195 PTAS.
(IVA Incluido)
MI COMPUTER
119
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
P.V.P. Canarias, Ceuta y Melilla 185 Ptas.
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen X-Fascículo 119

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 - Télex: 93392 EPPA

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 120 fascículos de aparición semanal, encuadernables en diez volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-183-6 (tomo 10)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 298604

Impreso en España-Printed in Spain- Abril 1986

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (120 fascículos más las tapas, guardas y transferibles para la confección de los 10 volúmenes) son las siguientes:

a) Un pago único anticipado de 27 105 ptas. o bien 10 pagos trimestrales anticipados y consecutivos de 2 711 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.





Puente informático

Concebido originalmente por Alan Kay, de Xerox, el futurista Dynabook se podría construir aplicando la tecnología de hoy día, aunque su precio lo convertiría en una propuesta poco práctica. En el futuro, sin embargo, un dispositivo como el que vemos podría convertirse en una ayuda esencial para todos, especialmente para aquellos grupos (la tercera edad, p. ej.) que están quedando cada vez más aislados en una sociedad tecnológica de cambios rápidos. Una pantalla redefinible sensible al tacto para la entrada, una visualización LCD de gran resolución, entrada de voz y una red de radio celular permitirían que el usuario controlara e interactuara con sistemas que irían desde informes meteorológicos locales hasta servicios bancarios personales. Puesto que el servicio se autofinanciaría, la unidad estándar podría suministrarse gratuitamente o bien bajo alquiler nominal del gobierno central

Un ordenador universal

Se conoce por Dynabook un conjunto de especificaciones ideales para un ordenador de ámbito universal formuladas en 1969

En 1969, un joven estudiante norteamericano de informática llamado Alan Kay presentó una tesis doctoral en la cual se imaginaba la invención de un ordenador portátil útil a nivel universal denominado Dynabook. El Dynabook habría de satisfacer todas las necesidades de proceso de información del usuario, incluyendo comunicaciones de audio y visuales y el acceso a bibliotecas de información pública. En 1969, el microprocesador no se había inventado aún y el tamaño de los ordenadores iba desde el de una gran nevera hasta el de varios armarios; lo que es más, las etiquetas de sus precios en dólares terminaban con al menos cinco ceros. El Dynabook de Kay era un auténtico sueño del futuro.

En 1971 Kay comenzó a trabajar en el Centro de Investigación de Xerox de Palo Alto (Palo Alto Research Centre: PARC) y ejerció su influencia en el grupo de investigaciones sobre el lenguaje (Language Research Group) que ideó el SMALLTALK, concebido originalmente como el lenguaje de programación del Dynabook y su sistema operativo. El equipo de Xerox nunca construyó el Dynabook: incluso tras la invención del microprocesador, en 1971, la tecnología existente sencillamente no era lo bastante potente. (Sin embargo, sí construyeron un ordenador portátil del tamaño de un maletín, unos cinco años antes de que Osborne hiciera renacer la idea.) No obstante, las concepciones que encerraba el SMALLTALK, tales como ventanas, iconos y ratones, poco a poco fueron filtrándose a la industria del ordenador, y finalmente condujeron al Lisa y el Macintosh de Apple y al Atari 520ST. Por su parte, Alan Kay trabaja actualmente como investigador para la Apple Corporation.

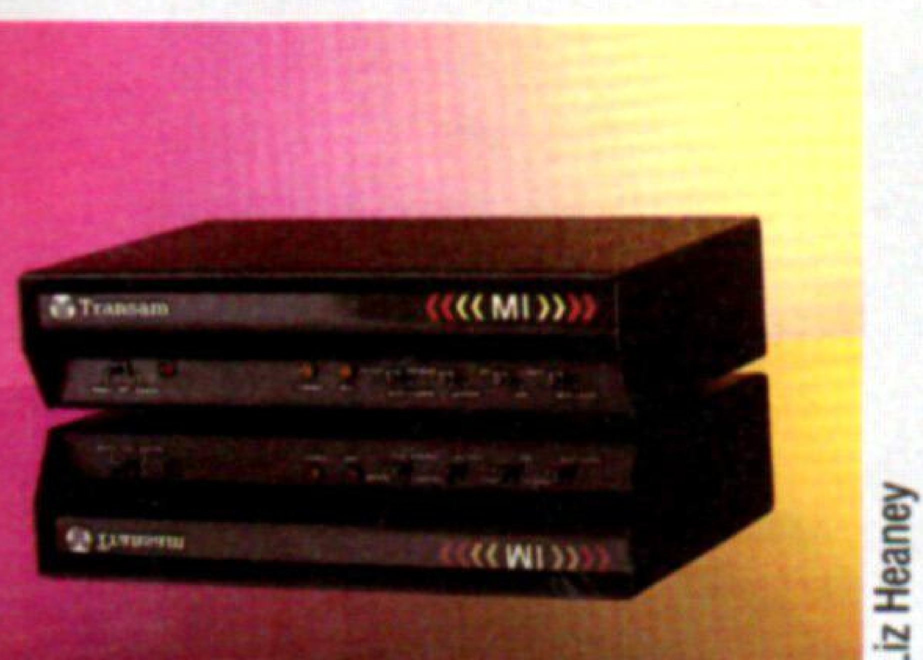
Una historia tan "antigua" como ésta es instructiva, porque el momento de construir el Dynabook está ya muy próximo: los componentes necesarios ya están disponibles o lo estarán muy pronto. Hoy, la visión de Alan Kay empieza a parecer viable.

El concepto original del Dynabook era un dispositivo del tamaño aproximado al de un libro encuadernado (*dyna[mic] book*: libro dinámico), y totalmente portátil gracias a su funcionamiento a pilas.

Requisitos tecnológicos

Demos una mirada a algunas de las tecnologías ya existentes o que están surgiendo y que podrían hacer realidad este ordenador del futuro.

- **Potencia de proceso:** Es necesario que en el centro de la máquina haya muchísima potencia de proceso. Ahora se pueden obtener en las tiendas microprocesadores de 32 bits con la potencia de un ordenador central en forma de chips individuales. La lista incluye al Motorola 68020, el Inmos T414 Transputer, el Intel 80386 y el Acorn ARM. Cada uno de ellos tiene capacidad de ejecutar entre tres y 10 millones de instrucciones por segundo.
- **Memoria:** Los fabricantes están produciendo chips de RAM de 256 Kbits, y son muchos los ordenadores personales que los incorporan. Pero todos están trabajando en la próxima generación de chips de un megabit, y algunos (incluyendo a la AT&T) en la actualidad hacen publicidad de componentes de muestra.
- **Visualización:** Varios fabricantes están produciendo visualizaciones de cristal líquido monocromáticas y planas que pueden simular la pantalla para gráficos de 640 × 200 del IBM PC. Ahora se están utilizando paneles electroluminosos para iluminar por detrás tales visualizaciones y salvar su principal inconveniente, es decir, su bajo contraste y sus limitados ángulos de visión. Además, la tecnología de la visualización plana se está desarrollando más rápidamente que casi todos los otros sectores de la industria, anunciándose nuevos desarrollos con una frecuencia casi semanal. Fabricantes japoneses, entre los que se incluye Epson, ya han mostrado prototipos de LED a color, mientras que Toshiba y otros están trabajando en tubos de plasma autoiluminados. El punto clave de la mayor parte de este trabajo de investigación es la búsqueda de una televisión de pantalla plana viable, que es lo que necesita el Dynabook. También habrá visualizaciones en pantalla plana sensibles al tacto, que probablemente eliminarán la necesidad del teclado. Una gran visualización sobre la superficie del ordenador incluiría una "imagen" sensible al tacto de un teclado, adaptable a medida bajo control de software. En cada etapa de un programa, se vería un conjunto diferente de "teclas" con etiquetas especiales, y se podrían dibujar imágenes directamente con un punzón en vez de con un ratón.
- **Fuente de alimentación:** El suministro de corriente eléctrica constituye el mayor problema. Las exigencias de las nuevas visualizaciones y las grandes memorias serían excesivas para las pilas de litio que se utilizan en los portátiles de regazo actuales, y la tecnología de la pila no está progresando necesariamente al mismo paso que el desarrollo de semiconductores. Sin embargo, poco a poco van surgiendo nuevos tipos de pilas con densidades de energía más elevadas. Y, al mismo tiempo, están declinando los requerimientos de potencia de los chips a medida que los fabricantes deciden aplicar la tecnología CMOS de bajo consumo.
- **Almacenamiento masivo:** Se cree que el disco láser compacto reemplazará con el tiempo a los diferentes tipos de almacenamiento magnético, tales como la cinta y los discos flexibles y Winchester. El Dynabook podría utilizar discos compactos de lectura como medio de distribución para poner en redes públicas "libros" de un interés muy especializado, música y publicaciones de video. Otra forma de almacenamiento masivo es la *tarjeta astuta*. Se trata de un dispositivo de memoria del tamaño de una tarjeta de crédito que se puede utilizar para todo, desde distribución de software hasta para llevar las cuentas bancarias; la empresa británica Cumana ahora distribuye software para ordenadores personales con su tarjeta Astron. En un futuro cercano, tales tarjetas podrían utilizarse para abonar mercancías y servicios a través del Dynabook
- **Sonido:** Durante un tiempo ha sido posible colocar todos los componentes de un sistema de audio en un único chip, un buen ejemplo de lo cual es el chip que contiene el Walkman Sony. Los receptores de radio y televisión se ven igualmente afectados (de allí la existencia de los televisores de bolsillo), y los chips de sintetizadores de música están progresando rápidamente. La voz, sin embargo, es un asunto diferente. Si bien la síntesis de voz es bastante fácil de obtener y los chips sintetizadores que se utilizan en la música moderna pueden realizarla a la perfección, la comprensión del lenguaje natural se encuentra aún en una etapa de desarrollo muy temprana.
- **Comunicaciones:** Las restantes piezas del rompecabezas del Dynabook pertenecen al área de las comunicaciones. Los modems telefónicos en un chip son ahora de uso común, pero no libertad de movimientos, el enlace se debe obtener por emisión radiofónica, no por cable. Ahora están comenzando a aparecer los primeros chips de modem de radio celular. El Dynabook se basa en una red de comunicaciones internacional de capas múltiples. Grandes ordenadores centrales contienen la información básica, y el primer paso para alcanzarlos es un enlace de radio celular con una estación local de ordenadores. Este nudo de red se comunica con los centros regionales mediante cables de fibra óptica con base en tierra, y éstos, a su vez, se comunican mediante más fibras ópticas y enlaces por satélite con los siguientes niveles de la jerarquía. Muchos de estos componentes ya se están instalando; lo que impide cualquier desarrollo ulterior es la inexistencia de una estandarización.



Liz Heaney

En el aire
La introducción de una red de radio celular ha estimulado el desarrollo de modems de radio celular. El modelo que vemos, el Transam M1, es un modem inteligente que ofrece facilidades de respuesta automática y llamada automática si se lo utiliza con un teléfono estándar. No obstante, también configura un conector de tipo D de 15 vías para acoplar a un transceptor, posibilitando el envío y la recepción de datos a través de la red de radio celular.

Incluiría una visualización de gráficos/televisión en color y sería capaz de presentar y procesar texto, imágenes y sonido. Sobre todo, sería un poderoso medio de comunicación.

Un ordenador auténticamente revolucionario no ha de ser sólo portátil, sino también capaz de intercambiar toda clase de datos con otros usuarios, proporcionándonos un medio preponderante de comunicación a larga distancia, como lo es ahora el teléfono. En resumen, podríamos transmitir una carta completa, con imágenes y sonido, y recibir una respuesta similar.

Pero esto no es más que el comienzo. Tendríamos acceso a diversos servicios públicos, desde bibliotecas, que nos proporcionarían libros y programas para ordenador que se cargarían en el sistema, así como noticias, información meteorológica, entretenimiento y facilidades educativas. También dispondríamos de informes sobre el estado de cuentas bancarias, comprobación de guías de calles, pedido de mercancías y reserva de billetes de viaje; todo ello desde cualquier punto e independientemente de las líneas telefónicas y los enchufes de la red eléctrica.

Si bien gran parte de la tecnología que se describe bajo el título *Requisitos tecnológicos* se halla en la vanguardia del desarrollo y, por lo tanto, es costosa, la financiación del Dynabook no constituye verdaderamente un problema; si se concretara de modo racional, sería el artículo de fabricación masiva definitivo (en cuanto a que todos y cada uno de los habitantes del mundo podría tener uno). Con tales volúmenes de producción, el costo de hardware por unidad sería prácticamente insignificante. Los problemas de marketing que tanto preocupan a los fabricantes de ordenadores personales de hoy en día se volverían bastante triviales si los ordenadores fueran realmente *útiles* para la gente: y el Dynabook, más que útil, sería indispensable.

Tampoco puede decirse que el mayor problema radique en la tecnología. Radica, más bien, en encontrar el incentivo para dar una forma concreta al sistema. En última instancia será un problema político, porque sólo los gobiernos poseen los recursos necesarios para tal cometido. Y, sin embargo, todo lo que haría el Dynabook ya se hace, con frecuencia a cargo de los estados, pero con sistemas distintos, ineficaces y caros. No es demasiado inverosímil imaginar una situación en la que el estado pudiera permitirse *regalar* Dynabooks, tal como está haciendo actualmente el gobierno francés con los terminales telefónicos de microordenador (son más baratos que imprimir y distribuir guías telefónicas de papel). Se pagaría más por los servicios que por el hardware, tal como ahora pagamos facturas de teléfono.



Punto final

Finalmente proporcionaremos el listado de ampliación de textos 6502 y un programa cargador en BASIC

Los usuarios del 6502 que no dispongan de ensamblador, o que deseen economizar tiempo de entrada, pueden colocar (POKE) las rutinas de compresión/ampliación de textos en la memoria utilizando el programa Cargador en BASIC 6502. Una vez ejecutado este programa, se puede borrar y sustituir por el programa Activador en BASIC. Éste es adecuado para usar tanto con las rutinas 6502 como con las versiones Z80 que hemos ofrecido previamente.

El programa Activador en BASIC le pide que entre la serie a comprimir y que especifique una dirección para los datos comprimidos a almacenar. Usted, por supuesto, deberá anotar esta dirección por si en una etapa ulterior quisiera recuperar los datos. Esto se realiza tan sólo especificando la dirección de los datos comprimidos de modo que resulten accesibles mediante la rutina de descompresión.

La rutina de compresión/descompresión Z80 se halla en la posición 30000, que resultará idónea para la mayoría de los micros. Si posee un ensamblador, puede reubicar el código simplemente cambiando la sentencia ORG. El programa Activador en BASIC se puede modificar para incluir sus propios programas, quizá para proporcionar facilidades para construir archivos más grandes de texto comprimido compuestos por registros de 255 bytes.

Ampliación de textos 6502

La siguiente rutina en lenguaje assembly se debe añadir a la rutina de compresión 6502 que ofrecíamos en la página 2346, insertándola antes de las tablas de datos finales. Esta rutina se ha escrito en el Commodore 64. Los usuarios del BBC Micro pueden convertirla introduciendo estos cambios:

- Cambiar las asignaciones de punteros de página cero, ZPTR1 a ZPTR4, a las direcciones &80 a &87.
- El formato de instrucción LABEL *=*+2 se debe cambiar por .LABEL EQUW
- El formato de instrucción LABEL *=*+1 se debe cambiar por .LABEL EQUB
- El formato de instrucción .BYT se debe cambiar por EQUS
- Las direcciones de llamada dependerán de dónde se ensamble la rutina. Las etiqs. START y EXPAND denotan las direcciones de llamada de las rutinas de compresión y ampliación y se acceden desde BASIC

```
;
;++++RUTINA AMPLIACION 6502++++
EXPAND  LDA  INPUT
        STA  ZPTR1
        LDA  INPUT+1     ;ESTABLECER PAGINA 0
        STA  ZPTR1+1     ;PUNTEROS PARA APUNTAR
        LDA  OUTPUT      ;HACIA ZONAS DE
        STA  ZPTR2       ;ENTRADA Y SALIDA
        LDA  OUTPUT+1
        STA  ZPTR2+1

        LDA  #255        ;INICIALIZAR
        STA  MASK        ;DESPLAZAMIENTOS
        LDA  #1          ;Y MASCARA
        STA  LEN         ;INIC DESPL. I/P
        STA  OUTOFF      ;INIC DESPL. O/P
;++AQUI EMPIEZA EL BUCLE PRINCIPAL++
NEXNIB  JSR  GETNIB
        BEQ  OUT8BT      ;NIBBLE 0 SIGNIFICA CODIGO 8BITS
        CMP  #2
        BEQ  EXIT        ;ES LA MARCA DE FINAL?
        BCC  OUTTOK      ;ES UN DISTINTIVO?
;++TRATAR CODIGO 4BITS++
        TAY              ;VALOR NIBBLE EN Y
        LDA  #<TAB4BT
        STA  ZPTR3       ;ESTABLECER ZPTR3 EN
        LDA  #>TAB4BT    ;COMIENZO DE TABLA
        STA  ZPTR3+1     ;DE 4 BITS
OUTCHR  LDA  (ZPTR3),Y   ;TOMAR CAR. DE LA TABLA
        LDY  OUTOFF      ;ESTABLECER Y PARA O/P
        STA  (ZPTR2),Y   ;ALMACENAR EN SERIE O/P
        INC  OUTOFF      ;BUMP PUNTERO DESPL. O/P
        JMP  NEXNIB      ;IR POR SIGUIENTE NIBBLE
;++TRATAR CODIGO 8 BITS++
OUT8BT  JSR  GETNIB      ;TOMAR NIBBLE CODIGO
        TAY              ;PONERLO EN Y
        LDA  #<TAB8BT    ;ESTABLECER ZPTR3 PARA
        STA  ZPTR3       ;APUNTAR AL COMIENZO
        LDA  #>TAB8BT    ;DE TABLA DE 8 BITS
        STA  ZPTR3+1
        JMP  OUTCHR      ;SACARLO
;++TRATAR DISTINTIVO++
OUTTOK  JSR  GETNIB      ;TOMAR PUNTERO DISTINTIVO
        STA  TABCNT      ;ALMACENARLO
        LDA  #<TOKTAB    ;ESTABLECER ZPTR3 PARA
        STA  ZPTR3       ;QUE MIRE AL COMIENZO
        LDA  #>TOKTAB    ;DE TABLA DE DISTINTIVOS
        STA  ZPTR3+1
        LDX  #0
        LDY  #0
TOKA    LDA  (ZPTR3),Y   ;TOMAR LONGITUD DE DISTINTIVO
        CPX  TABCNT      ;SI TENEMOS EL QUE QUEREMOS
        BEQ  FOUND       ;BIFURCACION
        CLC
        ADC  #1
        ADC  ZPTR3       ;ESTABLECER ZPTR3 EN COMIENZO
        STA  ZPTR3       ;DE SIGUIENTE DISTINTIVO
        LDA  ZPTR3+1
        ADC  #0
        STA  ZPTR3+1
        INX
        BNE  TOKA        ;ESTE SIEMPRE SE DEBE BIFURCAR!
FOUND   TAX              ;COLOCAR LONGITUD DISTINTIVO EN X
        LDY  #1          ;INICIALIZAR Y
FOUND1  LDA  (ZPTR3),Y   ;TOMAR CAR. DISTINTIVO
        STY  TEMP        ;ALMACENAR TEMP Y
        LDY  OUTOFF      ;TOMAR DESPL. O/P
        STA  (ZPTR2),Y   ;ALMACENAR CAR. EN SERIE O/P
        INC  OUTOFF      ;BUMP DESPL. O/P
        LDY  TEMP        ;VOLVER A TOMAR DESPL. DISTINTIVO
        INY
        DEX
        BNE  FOUND1      ;SI NO FINAL, TOMAR SIGUIENTE CAR.
        JMP  NEXNIB      ;RETOMAR BUCLE PRINCIPAL
;++TRATAR SALIDA++
EXIT    LDA  OUTOFF      ;TOMAR DESPL. O/P
        SEC
        SBC  #1          ;REDUCIR EN UNO
        LDY  #0          ;Y ALMACENAR AL PRINCIPIO
```

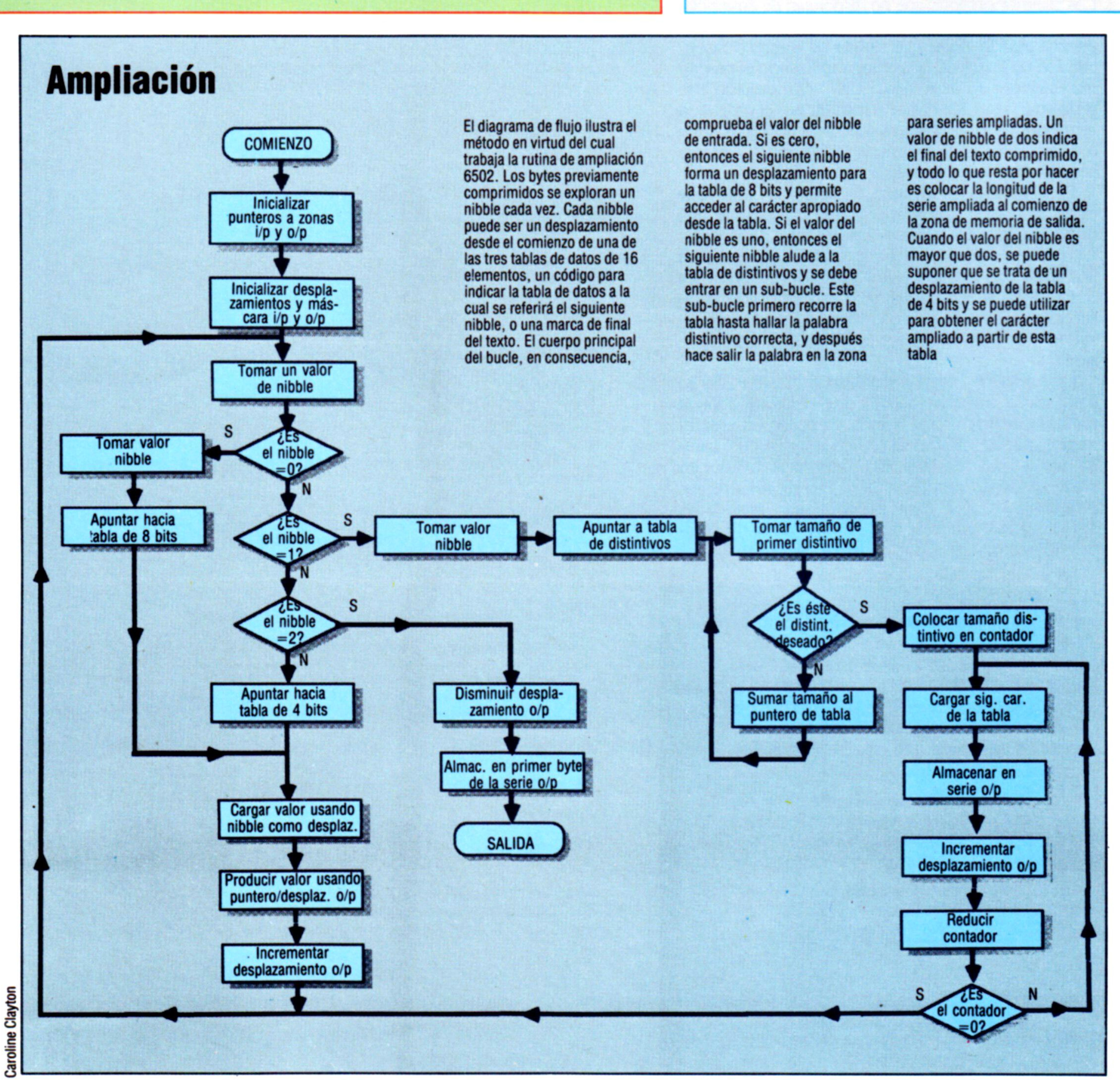
```
         STA   (ZPTR2),Y      ;DE SERIE O/P
         RTS                  ;RETORNAR AL SISTEMA
;++SUBRUTINA TOMAR UN NIBBLE++
GETNIB   LDA   MASK
         BNE   RIGHT          ;ES MASCARA 255?
         LDA   #255           ;PREPARAR MASCARA
         STA   MASK           ;PARA LA PROXIMA VEZ
         LDY   LEN            ;TOMAR DESPL. I/P
         LDA   (ZPTR1),Y      ;TOMAR BYTE I/P
         AND   #15            ;ENMASCARARLO
         INC   LEN            ;BUMP DESPL. I/P
         RTS
RIGHT    LDA   #0             ;PREPARAR MASCARA
         STA   MASK           ;PARA LA PROXIMA VEZ
         LDY   LEN            ;TOMAR DESPL. I/P
         LDA   (ZPTR1),4      ;TOMAR BYTE I/P
         LSR   A              ;DESPLAZAR NIBBLE IZQUIERDO
         LSR   A              ;HACIA LA MITAD DERECHA
         LSR   A              ;DEL BYTE
         LSR   A
         AND   #15            ;ENMASCARARLO
         RTS
```

Activador en BASIC Z80/6502

```
100 REM **********************************
110 REM ** Programa para comprimir/expandir **
120 REM **                                **
130 REM * una serie de 255 bytes (solo mayusc.) *
140 REM **********************************
150 CLS
160 INPUT " Version 6502 (S/N)";a$
170 IF a$="s" OR a$="S" THEN cf=1
180 INPUT "Expandir o Comprimir (E/C)?";a$
190 IF a$="" THEN COTO 180
200 IF a$="e" OR a$="E" THEN ec=1:GOTO 260
```

Ampliación

El diagrama de flujo ilustra el método en virtud del cual trabaja la rutina de ampliación 6502. Los bytes previamente comprimidos se exploran un nibble cada vez. Cada nibble puede ser un desplazamiento desde el comienzo de una de las tres tablas de datos de 16 elementos, un código para indicar la tabla de datos a la cual se referirá el siguiente nibble, o una marca de final del texto. El cuerpo principal del bucle, en consecuencia, comprueba el valor del nibble de entrada. Si es cero, entonces el siguiente nibble forma un desplazamiento para la tabla de 8 bits y permite acceder al carácter apropiado desde la tabla. Si el valor del nibble es uno, entonces el siguiente nibble alude a la tabla de distintivos y se debe entrar en un sub-bucle. Este sub-bucle primero recorre la tabla hasta hallar la palabra distintivo correcta, y después hace salir la palabra en la zona para series ampliadas. Un valor de nibble de dos indica el final del texto comprimido, y todo lo que resta por hacer es colocar la longitud de la serie ampliada al comienzo de la zona de memoria de salida. Cuando el valor del nibble es mayor que dos, se puede suponer que se trata de un desplazamiento de la tabla de 4 bits y se puede utilizar para obtener el carácter ampliado a partir de esta tabla



Caroline Clayton

```
210 IF a$="c" OR a$="C" THEN ec=0:GOTO 430
220 GOTO 180
230 REM
240 REM *** Expandir una serie comprimida ***
250 REM
260 PRINT "Donde esta la serie a ampliar?"
270 INPUT i
280 PRINT "Donde quiere la salida?"
290 REM Tenga cuidado de no machacar lenguaje
    maquina!
300 INPUT o
310 IF cf=0 THEN GOSUB 660:REM version Z80
320 IF cf=1 THEN GOSUB 700:REM version 6502
330 FOR x=1 TO PEEK(o)
340 PRINT CHR$(PEEK(o+x));
350 NEXT x
360 PRINT; "Continuar (s/n)? ":INPUT a$
370 IF a$ <> "S" AND a$ <> "s" THEN STOP
380 PRINT: GOTO 180
390 STOP
400 REM
410 REM ****** Comprimir una serie ******
420 REM
430 PRINT "Que he de comprimir? Recuerde que"
440 PRINT "La rutina solo acepta letras
    mayusculas"
450 PRINT
460 INPUT "Serie a comprimir: ";a$
470 PRINT "Donde debe ponerla?"
480 REM Ver advertencia de la linea 290!
490 INPUT i
500 PRINT "Donde quiere la salida?"
510 INPUT o
520 POKE i,LEN(a$)
530 FOR x=1 TO LEN(a$)
540 POKE i+x,ASC(MID$(a$,x,1))
550 REM poke i+x, code a$(x to x) en el
    Spectrum
560 NEXT x
570 GOSUB 660
580 PRINT "Longitud de la salida ";PEEK (o)
590 PRINT "Compresion conseguida: ";100-
    PEEK(o)/LEN(a$)*100;"%"
600 PRINT: "Continuar (s/n)? ":INPUT a$
610 IF a$<> "S" AND a$ <> "s" THEN STOP
620 PRINT: GOTO 180
630 POKE 30003,INT(i/256)
640 POKE 30002,i-PEEK(30003)*256
650 POKE 30005,INT(o/256)
660 POKE 30004,o-PEEK(30005)*256
670 RETURN
680 IF ec=1 THEN CALL 30270:RETURN
690 CALL 30000: IF PEEK (30006) <> 0 THEN PRINT
    "Compresion fallida": STOP
700 REM Version Commodore 6502
710 POKE 49153, INT (0/256)
720 POKE 49152, i-PEEK
730 POKE 49155, INT (i/256)
740 POKE 49154, i-PEEK(49155)*256
750 IF ec=1 THEN sys 49163
760 sys 49518: IF PEEK(49156)<>0 THEN
    PRINT"compresion fallida":STOP
770 RETURN
```

Cargador en BASIC 6502

```
10 REM ***********************************
15 REM ** CARGADOR EN BASIC COMPRESION **
20 REM ** Y EXPANSION COMMODORE 64      **
25 REM ***********************************
30 FOR I=49163 TO 49807:READ A:POKEI,A
35 CC=CC+A:NEXT I
40 READ CS:IF CS<> CC THEN PRINT "ERROR":STOP
100 DATA173,2,192,133,139,173,3,192
110 DATA133,140,173,0,192,133,141,173
120 DATA1,192,133,142,169,1,141,7,192
130 DATA160,0,177,139,141,6,192,169
140 DATA255,141,5,192,200,177,139,32
150 DATA39,193,208,54,32,129,192,240
160 DATA55,32,242,192,240,10,32,2,193
170 DATA72,169,0,32,65,193,104,32,65
180 DATA193,204,6,192,144,220,169,0
190 DATA141,4,192,169,2,32,65,193,173
200 DATA5,192,240,3,206,7,192,160,0
210 DATA173,7,192,145,141,96,169,255
220 DATA141,4,192,96,72,169,1,32,65
230 DATA193,104,32,65,193,76,79,192,72
240 DATA152,72,140,10,192,165,139,24
250 DATA109,10,192,133,251,165,140,105
260 DATA0,133,252,169,83,133,254,169
270 DATA194,133,255,169,0,141,8,192
280 DATA160,0,177,254,72,170,240,65
290 DATA165,254,24,105,1,133,254,165
300 DATA255,105,0,133,255,177,251,209
310 DATA254,208,22,200,202,208,246,104
320 DATA104,136,140,10,192,24,109,10
330 DATA192,168,104,173,8,192,162,0,96
340 DATA104,141,10,192,165,254,24,109
350 DATA10,192,133,254,165,255,105,0
360 DATA133,255,238,8,192,76,162,192
370 DATA104,104,168,104,162,255,96,72
380 DATA140,10,192,169,51,133,251,169
390 DATA194,133,252,104,76,15,193,72
400 DATA140,10,192,169,67,133,251,169
410 DATA194,133,252,104,160,0,209,251
420 DATA240,11,200,192,16,208,247,172
430 DATA10,192,162,255,96,152,172,10
440 DATA192,162,0,96,201,32,240,16,201
450 DATA44,240,12,201,46,240,8,201,65
460 DATA144,7,201,91,176,3,162,0,96
470 DATA162,255,96,72,140,10,192,172,7
480 DATA192,173,5,192,208,17,104,17
490 DATA141,145,141,238,7,192,172,10
500 DATA192,169,255,141,5,192,96,104
510 DATA10,10,10,10,145,141,172,10,192
520 DATA169,0,141,5,192,96,173,2,192
530 DATA133,139,173,3,192,133,140,173
540 DATA0,192,133,141,173,1,192,133
550 DATA142,169,255,141,5,192,169,1
560 DATA141,6,192,141,7,192,32,13,194
570 DATA240,28,201,2,240,106,144,37
580 DATA168,169,51,133,251,169,194,133
590 DATA252,177,251,172,7,192,145,141
600 DATA238,7,192,76,143,193,32,13,194
610 DATA168,169,67,133,251,169,194,133
620 DATA252,76,163,193,32,13,194,141,9
630 DATA192,169,83,133,251,169,194,133
640 DATA252,162,0,160,0,177,251,236,9
650 DATA192,240,16,24,105,1,101,251
660 DATA133,251,165,252,105,0,133,252
670 DATA232,208,233,170,160,1,177,251
680 DATA140,10,192,172,7,192,145,141
690 DATA238,7,192,172,10,192,200,202
700 DATA208,236,76,143,193,173,7,192
710 DATA56,233,1,160,0,145,141,0,173,5
720 DATA192,208,16,169,255,141,5,192
730 DATA172,6,192,177,139,41,15,238,6
740 DATA192,96,169,0,141,5,192,172,6
750 DATA192,177,139,74,74,74,74,41,15
760 DATA96,0,0,0,70,76,68,72,83,73,82
770 DATA78,79,65,84,69,32,67,77,85,71
780 DATA89,80,87,66,86,75,88,74,81,90
790 DATA44,46,3,84,72,69,4,84,72,73,83
800 DATA4,84,72,65,84,2,73,70,3,89,79
810 DATA85,2,77,69,3,87,65,83,2,72,69
820 DATA3,83,72,69,4,84,72,69,89,2,79
830 DATA70,2,73,84,2,73,83,3,70,79,82
840 DATA2,79,78,2,84,79,0,7
850 DATA76822:REM*SUMA DE CONTROL*
```

Una entrada

Por último, centramos nuestra atención en las facilidades a mayor escala que ofrece este sistema operativo

La tecnología del correo electrónico ya ha sido desarrollada desde hace un tiempo, si bien su implantación generalizada se ha visto condicionada en cierta medida por su relativa dificultad de aplicación, poca velocidad y falta de fiabilidad. El sistema de correo electrónico del Unix, el primer sistema operativo que lo incluyó como estándar, permite la comunicación entre usuarios de la misma máquina, y se puede ampliar para cubrir las comunicaciones entre dos ordenadores cualesquiera basadas en Unix, a través de un enlace adecuado. Veremos cómo funciona el sistema entre usuarios de la misma máquina; puesto que la operatoria es idéntica para usuarios remotos, resultará una introducción idónea para ambos casos.

El sistema se activa mediante la instrucción mail, que abre su archivo de correspondencia personal y proporciona acceso a toda la correspondencia que le haya sido enviada. Todos los mensajes entrantes se identifican por un número y el "ID" del remitente. Si se enviaran copias a otros usuarios, también se indicaría este hecho. Los mensajes nuevos y sin leer llevan un indicador (cada uno normalmente va acompañado por un título de una línea para identificar el tema de que se trate) y se pueden leer ya sea individualmente o bien por el orden en que llegaron con sólo pulsar la tecla Return.

El sistema se identifica mediante el aviso &, que posee una variedad de comandos relativos a la manipulación del correo y las funciones del sistema, como cambiar el directorio, y usted puede salir con el comando x o q. El primero deja en el buzón toda la correspondencia que no se haya eliminado, de modo que volverá a aparecer la próxima vez que se active el sistema; el segundo extraerá los mensajes que el usuario ya ha leído y los colocará en su directorio bajo un archivo especial llamado mbox. En este punto están disponibles para la edición y las

Diálogo con la base de datos

```
Berkeley 4.2 Vax/Unix (infc3)
Type (Ctrl-D) to disconnect

login: com–mcc
Password:
You are a normal user (class 3)
Jobs: 6 Superiors: 2 Maximum: 21
Last login: Thu Nov 14 13:47:42 on ttyn04

You have mail.                    {se le informa si tiene corresp. aguardando}

%mail
Mail version 2.18 5/19/83. Type ? for help.
"/usr/spool/mail/com–mcc": 1 message 1 new
>N 1 com–vjp Fri Nov 15 09:52 11/271 "Specimen mailing"
&                                 {pulse sólo RETURN para obtener la nueva
                                  corresp.}
Message 1:
From com–vjp Fri Nov 15 09:52:35 1985
From: com–vjp (Vicki)
To: com–daf, com–mcc
Subject: Specimen mailing         {título del mensaje}
Hello                             {contenido del mensaje}

&?                                {da una lista de comandos}
     Mail          Commands
t    <message list>        types messages
n                          goto and type next message
e    <message list>        edit messages
f    <message list>        give head lines of messages
d    <message list>        delete messages
s    <message list> file   appends messages to files
u    <message list>        undelete messages
r    <message list>        reply to messages
pre  <message list>        make messages go back to/usr/mail
m    <user list>           mail to specific users
q                          quit, saving unresolved messages in mbox
x                          quit, do not remove system mailbox
h                          print out active message headers
!                          shell escape
c    [directory]           chdir to directory or home if none given
A    <message list>        consists of integers, ranges of same, or user names
                           separated by spaces. If omitted, Mail uses the last
                           message typed.
A    <user list>           consists of user names or distribution names separated
                           by spaces.
Distribution names are defined in .sendrc in your home directory.

% ingres demo                     {ahora ejecute la base de datos "ingres" utilizando datos
                                  de demo}
INGRES version 7.10 (10/27/81) login
Fri Nov 15 10:28:42 1985
go
* print parts                     {comandos entrados en el espacio de trabajo y activadas
                                  mediante \g}
*\g
Executing...

parts relation
|pnum |pname              |colour |weight |qoh |
|-----------------------------------------------|
|  10 |byte-soap          |clear  |     0 | 143|
|   1 |central processor  |pink   |    10 |   1|
|  11 |card reader        |grey   |   327 |   0|
|   2 |memory             |grey   |    20 |  32|
|  12 |card punch         |grey   |   427 |   0|
|   3 |disk drive         |black  |   685 |   2|
|  13 |paper tape reader  |black  |   107 |   0|
|   4 |tape drive         |black  |   450 |   4|
|  14 |paper tape punch   |black  |   147 |   0|
|   5 |tapes              |grey   |     1 | 250|
|   6 |line printer       |yellow |   578 |   3|
|-----------------------------------------------|
continue
*\p                               {muestra el contenido actual del espacio de trabajo}
print parts
```

otras funciones normalmente relacionadas con el tratamiento de archivos de texto, pero el sistema de correspondencia no puede acceder directamente a ellas.

Usted puede enviar correspondencia desde el sistema mediante el comando m, o desde fuera del mismo impartiendo el comando mail seguido del nombre (o nombres) del destinatario. Se utilizan los ID de *login* normales y la lista de nombres puede ser tan larga como desee el usuario; cada uno recibirá una copia de su mensaje.

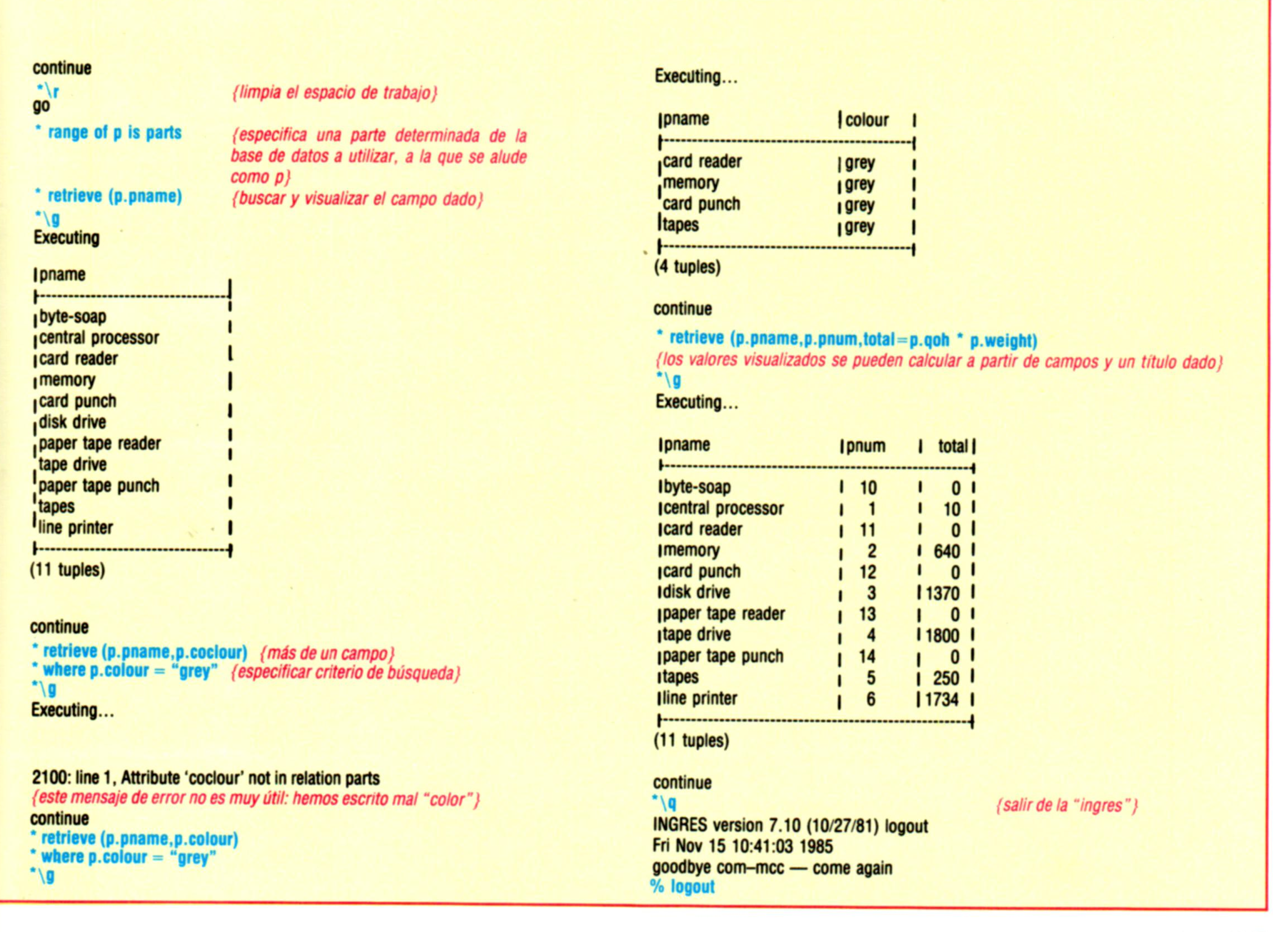
Cuando la correspondencia se distribuye regularmente entre varios usuarios, usted puede ahorrarse el tener que escribir una lista de nombres cada vez merced a la facilidad alias. Para estos alias se utiliza un archivo especial, así como para establecer otros parámetros de correspondencia, como pedir que se le informe cuándo ha llegado algún mensaje (utilizando ya sea .mailrc o bien .sendrc).

La base de datos "ingres"

Con la mayoría de los sistemas Unix se suministra, o al menos está disponible, la base de datos *ingres*. Si bien su interface para el usuario es menos fácil de aplicar que la mayoría, es una base de datos relacional muy potente, muchísimo más, de hecho, que la mayoría de los sistemas más conocidos, como el dBase II. Pero así como la mayoría de los aspectos del Unix, se la puede adaptar a la medida para satisfacer sus propias necesidades si así fuera necesario, un ejemplo de lo cual podemos apreciar abajo.

Ningún sistema está completo sin alguna forma de editor de textos: el Unix proporciona al menos tres como estándar y opcionalmente se pueden añadir otros. El más básico es ed, un editor de línea similar tanto al ed del CP/M como al edline del MS-DOS. Al igual que todos los editores de línea, es más bien difícil de utilizar, pero posee la ventaja de tener todas las funciones de edición como comandos. Por consiguiente, se puede preparar un archivo que contenga una secuencia de comandos, permitiendo que el Unix lleve a cabo automáticamente la edición de un documento. Los otros dos editores son ded y vi, que se pueden utilizar como editores tanto de pantalla como de línea.

Por último, por razón de espacio nos hemos visto limitados a cubrir sólo una fracción de las facilidades del Unix, pero las que se han omitido están relacionadas mayormente con la programación y el desarrollo de programas, que interesan más a los usuarios avanzados que a los recién iniciados. El Unix es, indudablemente, uno de los sistemas operativos más importantes desde el punto de vista del desarrollo de la informática moderna, y aunque quizá jamás consiga un éxito comercial aplicado en los micros, su influencia se puede apreciar con toda claridad en casi todos los nuevos sistemas.



```
continue
*\r                          {limpia el espacio de trabajo}
go
* range of p is parts        {especifica una parte determinada de la
                             base de datos a utilizar, a la que se alude
                             como p}
* retrieve (p.pname)         {buscar y visualizar el campo dado}
*\g
Executing

|pname                  |
|-----------------------|
|byte-soap              |
|central processor      |
|card reader            |
|memory                 |
|card punch             |
|disk drive             |
|paper tape reader      |
|tape drive             |
|paper tape punch       |
|tapes                  |
|line printer           |
|-----------------------|
(11 tuples)

continue
* retrieve (p.pname,p.coclour)  {más de un campo}
* where p.colour = "grey"   {especificar criterio de búsqueda}
*\g
Executing...

2100: line 1, Attribute 'coclour' not in relation parts
{este mensaje de error no es muy útil: hemos escrito mal "color"}
continue
* retrieve (p.pname,p.colour)
* where p.colour = "grey"
*\g

Executing...

|pname              |colour |
|---------------------------|
|card reader        |grey   |
|memory             |grey   |
|card punch         |grey   |
|tapes              |grey   |
|---------------------------|
(4 tuples)

continue
* retrieve (p.pname,p.pnum,total=p.qoh * p.weight)
{los valores visualizados se pueden calcular a partir de campos y un título dado}
*\g
Executing...

|pname              |pnum  | total|
|---------------------------------|
|byte-soap          | 10   |    0 |
|central processor  |  1   |   10 |
|card reader        | 11   |    0 |
|memory             |  2   |  640 |
|card punch         | 12   |    0 |
|disk drive         |  3   | 1370 |
|paper tape reader  | 13   |    0 |
|tape drive         |  4   | 1800 |
|paper tape punch   | 14   |    0 |
|tapes              |  5   |  250 |
|line printer       |  6   | 1734 |
|---------------------------------|
(11 tuples)

continue
*\q                                        {salir de la "ingres"}
INGRES version 7.10 (10/27/81) logout
Fri Nov 15 10:41:03 1985
goodbye com–mcc — come again
% logout
```

Suave transición

Finalmente examinaremos el Mikro de Supersoft, un ensamblador/monitor basado en cartucho para el Commodore 64

Entre los problemas que usted deberá afrontar cuando escriba por primera vez programas en código máquina se incluyen la falta de "protección" de la máquina inherente a programación en lenguaje máquina y el nada familiar método de preparar programas para su ejecución. Lo primero es el precio que el usuario ha de pagar por la potencia que le confiere la programación en lenguaje máquina.

El cartucho ensamblador Mikro de Supersoft permite entrar texto de programas fuente utilizando un editor de pantalla completa y números de línea, como los que emplea el BASIC del Commodore 64. Puesto que la mayoría de las personas que aprendan lenguaje máquina en su micro habrán primero aprendido BASIC, este método de entrada les resultará familiar y les ayudará a suavizar la transición de la programación en BASIC al trabajo en lenguaje máquina.

Supersoft Mikro

Mikro, de Supersoft, es desde hace tiempo el estándar para el Commodore 64. Cuenta con la ventaja de estar alojado en un cartucho enchufable, proporcionando un ensamblador y monitor completos integrados con el BASIC Commodore. Los programas fuente se entran como líneas normales de un programa en BASIC; el usuario no puede ejecutar el programa (RUN), pero sí utilizar las instrucciones LOAD y SAVE normales y el editor en pantalla de BASIC para almacenar y corregir su código fuente. Aunque ésta es una solución discutible, el Mikro añade una instrucción de números de línea AUTOmáticos, una instrucción DELETE para suprimir líneas y una serie FIND para facilitar la edición. Falta la instrucción esencial RENUMBER, y si usted ha de renumerar su programa habrá de cargar una utilidad separada. Durante el ensamblaje, el Mikro puede producir un listado assembly completo en una impresora, pero no en pantalla, y también una lista clasificada de los símbolos utilizados en los programas. Las directivas de ensamblador son adecuadas, y la más útil es LNK, que permite escribir un programa en secciones y luego ensamblarlas desde cassette o disco como un programa. Ello facilita el desarrollo de programas de hasta unos 12 Kbytes de código objeto.

Liz Heaney

El Mikro posee un monitor de lenguaje máquina similar a muchos existentes para el C64 y a aquellos incorporados en la gama Commodore PET. Esto es práctico, puesto que no incluye puntos de parada ni seguimiento. Se entrega, además, un breve manual de 16 páginas, que, aunque describe los comandos, no es todo lo detallado que debería ser. En resumidas cuentas, el Mikro es adecuado y fiable, pero puede resultar incómodo para desarrollar programas largos.

Tipo de ensamblador

Dos pasos, mnemotécnicos estándar

Límites

Código fuente: 30 K, pero puede enlazar archivos
Tamaño de la tabla de símbolos: 11 K

Directivas de ensamblador

*	Se utiliza en lugar de ORG y DEFS. Alude a la posición actual, y una instrucción como *=$6000 indicará al ensamblador que empiece el progra[illegible] la posición $6000. Una instrucción como =* +256 reservaría un espacio de 256 bytes para datos en el programa
=	Asignar un valor a un símbolo
WOR	Almacenar valor(es) de 16 bits
BYT	Almacenar valor(es) de 8 bits
TXT	Almacenar serie ASCII
LNK	Continuar ensamblando desde el arch. fuente mencionado
OUT	Sacar listado assembly por impresora
OFF	Desactivar listado

Opciones de ensamblaje

Ninguna

Aritmética

+,=,< y > (bytes *high* y *low* de valores de 16 bits). Decimal, hexa y binario



En los años noventa

Traslademonos a principios de la próxima década e imaginemos un ordenador que reúna las características que se intuye incluirán las máquinas de ese cercano futuro

Los primeros años de la década de los noventa han sido testigos de cómo numerosas características que antes sólo estaban disponibles en micros de gestión se incorporaban en los micros personales. Máquinas de interface amables con el usuario acopladas con una creciente cantidad de facilidades a disposición del aficionado al ordenador personal han hecho que el ordenador moderno sea fácil de usar y le han conferido facilidades en la propia placa para una amplia gama de aplicaciones.

El Chestnut ("castaña"), de la firma Conquer, es la última producción de una línea de "puestos de trabajo personales" fabricada para el inmenso mercado del micro doméstico. Ahora que, al fin, se está comprendiendo el verdadero potencial del ordenador personal (recientes encuestas han indicado que la base de usuarios es ahora equivalente a la de los sistemas de alta fidelidad), el mercado está siendo "activado por la demanda" en vez de por la "oferta", es decir, los fabricantes están atendiendo las necesidades de los usuarios en lugar de producir artilugios de alta tecnología con los que confían en hallar un mercado.

A alrededor de £500 (o sea, poco más de 100 000 ptas), el Chestnut incluye las facilidades más populares de los últimos cinco años. Basado en el conocido procesador 68000, el paquete incluye un ordenador con pantalla en color integral, una única unidad de disco de 3½ pulgadas, doble densidad y doble cara, un minidisco rígido de 10 Mbytes y un megabyte de RAM. Midiendo apenas 300 × 240 × 120 mm, la carcasa del ordenador se hace eco de la reciente tendencia a crear máquinas planas cada vez más pequeñas y más en armonía con la superficie del escritorio.

El teclado y el mando de bola/ratón están conectados al ordenador a través de cables de fibra óptica. El teclado responde al formato QWERTY estándar con 10 teclas de función y un teclado numérico que también actúa como grupo de cursor. El teclado de máquina de escribir contiene varias teclas de función, como Ctrl y Alt, que permiten programar en la máquina funciones adicionales.

Las dimensiones del Chestnut le deben mucho a la tecnología de "pantalla plana" que se ha perfeccionado de manera notable en el curso de los últimos años. La pantalla no sólo mide la mitad que las pantallas de tubos de rayos catódicos de los años setenta y ochenta, sino que en los bordes ya no se produce distorsión de imagen.

Puerta en paralelo

La CD-ROM se considera como un periférico esencial. Al objeto de ser competitivo, el Chestnut se proporciona con una adecuada puerta en paralelo. Esta primera unidad de CD-ROM de Hitachi retenía 552 Mbytes de datos y su velocidad de transferencia de información era de 176 Kbytes por segundo



Liz Heaney

Debajo de la pantalla se halla la unidad de disco, que proporciona un disco individual de doble cara con capacidad de un megabyte. Opcionalmente se puede instalar una segunda unidad en un espacio ahora tapado por el logotipo del Chestnut, justo debajo de la primera unidad (Conquer pretende comercializar un modelo de unidades gemelas para aplicaciones de gestión).

Sobre el lado derecho de la carcasa principal del ordenador está la puerta controladora del ratón. Asimismo, hay un bus de ampliación en donde se pueden instalar segundos procesadores, procesadores de fotogramas y otros módulos de ampliación de memoria de un megabyte.

El panel posterior de la máquina alberga las puertas estándares para periféricos y comunicaciones. En el extremo derecho hay un par de conectores de *hi-fi* en los que se puede enchufar un sistema estéreo. Ello permite sacar el máximo partido del soberbio sonido del Chestnut, proporcionado por el chip sintetizador Y/S2416 que posee ocho osciladores en la placa y una gama de cinco octavas. De

este modo se puede producir un sonido polifónico con un teclado de piano electrónico.

La puerta para el usuario, que da cabida a teclados musicales o dispositivos robóticos, se halla a continuación, seguida por los conectores en serie MIDI. Aunque actualmente la mayoría de los músicos están utilizando el formato (en paralelo) MIDI-P, Conquer ha decidido incluir en el Chestnut las puertas en serie MIDI antiguas, para sacar partido de la gran cantidad de paquetes LAN producidos para el estándar MIDI-S.

Por supuesto, ya no hay necesidad de programar el chip sintetizador directamente. Puesto que ahora el software MIDI se proporciona en placa, la programación del chip es simplemente cuestión de configurar el sintetizador como un dispositivo MIDI y enviarle los códigos adecuados.

Al igual que la mayoría de los micros modernos, todas las conexiones en serie para la MIDI y el modem en placa se hallan en un único chip. El acceso a la red telefónica lo proporciona un par de enchufes telefónicos estándar, que permiten al usuario acceder a toda la gama de comunicaciones disponibles. Éstas incluyen Micronet, Final Frontier (el juego recreativo para múltiples usuarios) y Telemarket, un servicio en virtud del cual se pueden adquirir programas de juegos y aplicaciones a través de la red telefónica y cargarlos en el disco rígido del usuario.

La nueva generación de micros personales incorpora una facilidad que, comprensiblemente, ha entusiasmado al público: un fiable reconocimiento y síntesis de voz. Esta característica no ha derivado de innovación tecnológica alguna, sino de la drástica caída del precio de la memoria en los últimos años y de la creciente disponibilidad de procesadores de elevado rendimiento. Ello ha vuelto disponible memoria suficiente para almacenar las plantillas de voz que se requieren para el proceso.

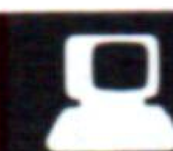
Un micrófono enchufado en el conector hembra D/A de la parte posterior del Chestnut le transmitirá instrucciones habladas al ordenador, el cual buscará las plantillas para hallar una pareja y responder, una vez encontrada ésta. Mediante el uso de estas mismas plantillas y el chip sintetizador, el Chestnut alcanza un elevado nivel de síntesis de voz y una limitada capacidad para conversar.

Junto a la interface D/A se halla la puerta para impresora Centronics habitual, y sobre el extremo izquierdo hay una interface en paralelo para una unidad de CD-ROM. Conquer cree que éste es uno de los más importantes aciertos de la máquina, dado que los usuarios podrán acceder a la gran cantidad de material de bases de datos, tales como la *Enciclopedia británica*, que recientemente ha aparecido en CD-ROM. Los ávidos aventureros podrán jugar a la popular *Odyssey*, de 500 Mbytes, que saca máximo partido de esta tecnología: el acceso rápido al almacenamiento masivo ha permitido a los programadores crear una aventura completamente activada por la voz y con detallados gráficos animados.

La pantalla de gran resolución que acompaña al Chestnut puede visualizar más de mil colores a la vez. El chip *blitter* de video proporciona una visualización uniforme y realista en modalidad Animation.

El WIMPOS-3, sistema operativo activado por menú, es similar a anteriores versiones desarrolladas por Conquer. La única diferencia parece residir en su mayor alcance para la manipulación de ventanas e iconos a través de instrucciones habladas, si bien el modelo que probamos en ocasiones no resultó fiable, especialmente cuando se le hablaba pronunciando a una velocidad más cercana a la del habla normal. Al menos por ahora, parecería que el uso del ratón, aunque anticuado, es preferible a correr el riesgo de perder archivos a raíz de una mala interpretación por parte de la máquina de las instrucciones vocalizadas.

El software empaquetado en el disco rígido del



Conquer Chestnut

DIMENSIONES
300 × 240 × 120 mm

CPU
68000 operando a 8 MHz

MEMORIA
1 Mbyte de RAM; 128 Kbytes de ROM

PANTALLA
Resolución de textos de 80 × 32; resolución de gráficos de 640 × 256 pixels, permitiendo visualizar simultáneamente hasta mil colores

INTERFACES
Bus de ampliación, conector para ratón/mando de bola, conectores *hi-fi* estéreo, puerta para el usuario, puertas MIDI gemelas, conectores red telefónica gemelos, conector A/D, puerta CD-ROM en paralelo

LENGUAJES DISPONIBLES
BASIC, compilador de C

UNIDAD DE DISCO
Individual, de 3½ pulgadas, doble cara y doble densidad, de un Mbyte, con una segunda unidad opcional

SISTEMA OPERATIVO
WIMPOS-3

DOCUMENTACION
Es completa y ofrece información pormenorizada acerca del sistema. Como detalle original, Conquer ofrece los manuales en CD-ROM, así como en la versión impresa normal

PUNTOS FUERTES
El Chestnut ofrece numerosas facilidades que hasta muy recientemente sólo estaban disponibles en máquinas más caras

PUNTOS DEBILES
La máquina no ofrece nada radicalmente nuevo y quizá experimente problemas para hacer oír su voz entre los productos de "tecnología punta" existentes en el mercado

ordenador incluye el programa de tratamiento de textos *ChestWrite*, el completísimo *ChestPaint*, el *ChestTalk*, para desarrollar plantillas de voz definidas por el propio usuario, así como un sistema experto, ChestComplaint, que permite que el mismo usuario diagnostique su estado de salud. Asimismo, Conquer suministra con la máquina el sistema WIMPOS, con el software de modem y MIDI. Los lenguajes de programación empaquetados con el Chestnut son BASIC y un compilador de C.

A pesar de las reiteradas afirmaciones de Conquer durante los seis meses que precedieron al lanzamiento de la máquina, el Chestnut ofrece pocas novedades. Su principal atractivo comercial probablemente residirá en su bajo precio y sus facilidades empaquetadas, las cuales, en las anteriores máquinas de la empresa, se vendían como extras opcionales. Pero a la vista de que los críticos informáticos han alcanzado tal talla que se han puesto a la altura de sus homólogos en el mundo del cine, el teatro y la literatura, resulta demasiado sencillo juzgar con severidad las nuevas máquinas de hoy en día, que hace apenas siete u ocho años habrían representado la cima de los logros de la informática.





Factores decisivos

Al elegir un lenguaje de programación es preciso armonizar las necesidades del usuario con las facilidades de aquél

Hasta hace poco tiempo, los usuarios de micros pequeños sólo disponían de BASIC o de lenguaje assembly. Ahora incluso las máquinas más pequeñas disponen de la mayoría de los lenguajes comunes, y usted sólo tiene que pasar apenas a una máquina compatible con el IBM PC o a cualquier otra máquina MS-DOS para disponer virtualmente de una gama completa. Dado que los compiladores e intérpretes pueden ser bastante caros y que, por lo tanto, pocas personas desearán tener más de tres o cuatro (si acaso), el problema de la elección subsiste a pesar de la mayor disponibilidad.

La eficacia, desde el punto de vista de la memoria utilizada y los tiempos de ejecución, puede convertirse en un factor importante, particularmente con programas que estén interactuando con el entorno externo. En BASIC, por ejemplo, algunos programas llevarían tanto tiempo de escribir y depurar, que sería más rápido aprender un lenguaje más adecuado y escribirlo en el mismo. Un programa para interrogar y actualizar un archivo de existencias de artículos en línea, por ejemplo, sería una tarea sencilla en COBOL empleando archivos indexados. Escrito en BASIC, habría de utilizar archivos se-

Lo bueno y lo malo

Lenguaje: BASIC
Puntos positivos: Fácil de aprender y utilizar, ampliamente disponible. Barato. Buena provisión de funciones aritméticas. Buen tratamiento de series
Puntos negativos: La mayoría de las versiones no poseen una buena facilidad de módulos ni estructuras de control. No estandarizado. La ejecución a menudo es ineficaz y lenta. Tratamiento de archivos pobre. Gama restringida de tipos de datos y estructuras. Fácil de escribir mal, código no claro
Uso: Programas cortos, de usar y tirar si se trata de problemas que impliquen aritmética o tratamiento de series

Lenguaje: LOGO
Puntos positivos: Sólida base matemática, fácil de aprender a un nivel básico. Gráficos de tortuga. Buenas facilidades para proceso de listas, buena provisión de módulos, tipos de datos y estructuras. Disponible amplia y económicamente
Puntos negativos: La programación más allá de la etapa inicial puede volverse complicada y críptica. Muchas versiones incompatibles. Ejecución ineficaz a veces
Uso: Gráficos, proceso de listas, aprendizaje sobre matemáticas y conceptos de programación avanzada

Lenguaje: PASCAL
Puntos positivos: Bien estructurado. Buena gama de tipos de datos. Relativamente estandarizado. Es fácil producir código de gran calidad
Puntos negativos: Entrada/salida no definidas con claridad, el tratamiento de archivos es deficiente
Uso: Aprendizaje de buenas técnicas de programación. Programación general semimatemática a pequeña escala

cuenciales o un algoritmo de *hashing* y archivos de acceso directo que lo volverían muchísimo más complicado de lo que sería necesario.

Las características de diseño que hacen que un lenguaje sea fácil de aprender y utilizar también tienden a crear problemas cuando aumenta el tamaño del programa, circunstancia en la cual uno de los requisitos fundamentales debe ser la facilidad para descomponer el programa en módulos moderadamente autocontenidos que se puedan programar independientemente.

Algunos lenguajes, como el COBOL y el FORTRAN, están definidos rígidamente por comités internacionales, lo que convierte en un lento procedimiento la introducción de cualquier cambio para reflejar el hardware y los nuevos tipos de proceso. Sin embargo, un programa escrito en una máquina sólo requerirá una recompilación, o, en el mejor de los casos, unos pocos cambios menores, para ejecutarse en una máquina diferente.

Los lenguajes como el PASCAL y el C poseen un estándar *de facto* definido por su(s) inventor(es). La mayoría de las versiones de estos lenguajes son bastante coherentes con el estándar, pero, especialmente en lenguajes como el PASCAL, en el cual la entrada/salida no está definida claramente, cada implementador tiene la libertad de introducir modificaciones y adiciones, aunque los programas que utilicen estas facilidades no serán muy portables. Por último, tenemos lenguajes como el BASIC, para el cual no existe estándar ninguno.

Otro factor es el tipo de programa traductor utilizado. Los intérpretes tienden a ser fáciles de emplear y mejores para el desarrollo de programas, pero más lentos y menos eficaces en ejecución. Los compiladores son más complicados de utilizar, pero producen un producto final más eficaz. Esto ahora se está reduciendo por la introducción de compiladores incrementales y depuradores animados.

Algunos lenguajes han suscitado desarrollos de hardware para permitir una ejecución más eficaz. Por ejemplo, se ha dedicado muchísimo esfuerzo en producir un procesador que ejecute FORTH directamente. Del mismo modo, existe un procesador que ejecuta P-code PASCAL. En el futuro cercano quizá sea posible producir un compilador basado en hardware, que simplificará mucho el proceso de utilizar un lenguaje compilado.

No obstante, el factor primordial continúa siendo el tipo de aplicación para la cual se esté utilizando el lenguaje. Veamos algunos lenguajes y comparemos sus puntos fuertes y sus puntos débiles. Luego podemos ver algunas de las aplicaciones típicas y considerar qué lenguaje sería el más adecuado.

Lenguaje: FORTH
Puntos positivos: Ejecución eficaz, ampliable por el usuario. Buenas estructuras
Puntos negativos: Nivel demasiado bajo para gran parte de trabajo serio. Difícil de comprender. Muchas versiones diferentes
Uso: Trabajo de bajo nivel antes que el ensamblador, especialmente hardware controlador

Lenguaje: FORTRAN
Puntos positivos: Muy estandarizado. Grandes bibliotecas de software disponibles. Excepcional gama de funciones matemáticas y científicas
Puntos negativos: Anticuado. Estructuras pobres. Es fácil escribir código críptico. Entrada/salida diseñada en base a tarjetas perforadas. Trazado de programas estricto
Uso: Programación general científica, de ingeniería y matemáticas

Lenguaje: COBOL
Puntos positivos: Sumamente estandarizado, disponible en una gama de máquinas muy amplia. Fácil de aprender. Excelente tratamiento de archivos
Puntos negativos: Verborreico, difícil de aprender Los compiladores son grandes y caros. Mal estructurado, con una pobre gama de tipos de datos
Uso: Proceso de datos en gestión

Lenguaje: C
Puntos positivos: Buena gama de tipos de datos y estructuras. Buena estructura modular. Proporciona acceso al hardware. Produce código veloz y eficiente
Puntos negativos: Resulta fácil producir código críptico. Nivel demasiado bajo para muchos trabajos serios. Estandarizado inadecuadamente, basándose demasiado en las bibliotecas
Uso: Software de sistemas como recambio para el ensamblador. Allí donde se requiera máxima eficacia

Lenguaje: LISP
Puntos positivos: Proceso de listas matemáticamente sólido. Mucho software y apoyo disponibles. Muy utilizado en las máquinas grandes
Puntos negativos: Difícil de aprender, comprender y utilizar
Uso: Inteligencia artificial y proceso de listas

Lenguaje: PROLOG
Puntos positivos: Sólida base matemática. Fácil y entretenido de utilizar. Supuestamente más próximo al razonamiento humano
Puntos negativos: No es un lenguaje relacional "puro": posee algunos aspectos procedurales y necesita algunas facilidades extrañas, como el corte, para operar eficientemente
Uso: Inteligencia artificial, aplicaciones de bases de datos

Lenguaje: Ensamblador
Puntos positivos: Control absoluto de todos los aspectos. Potencialmente el más eficiente
Puntos negativos: Completamente no estándar. Difícil de aprender y utilizar
Uso: Máxima eficiencia, pero sólo si fallan los demás

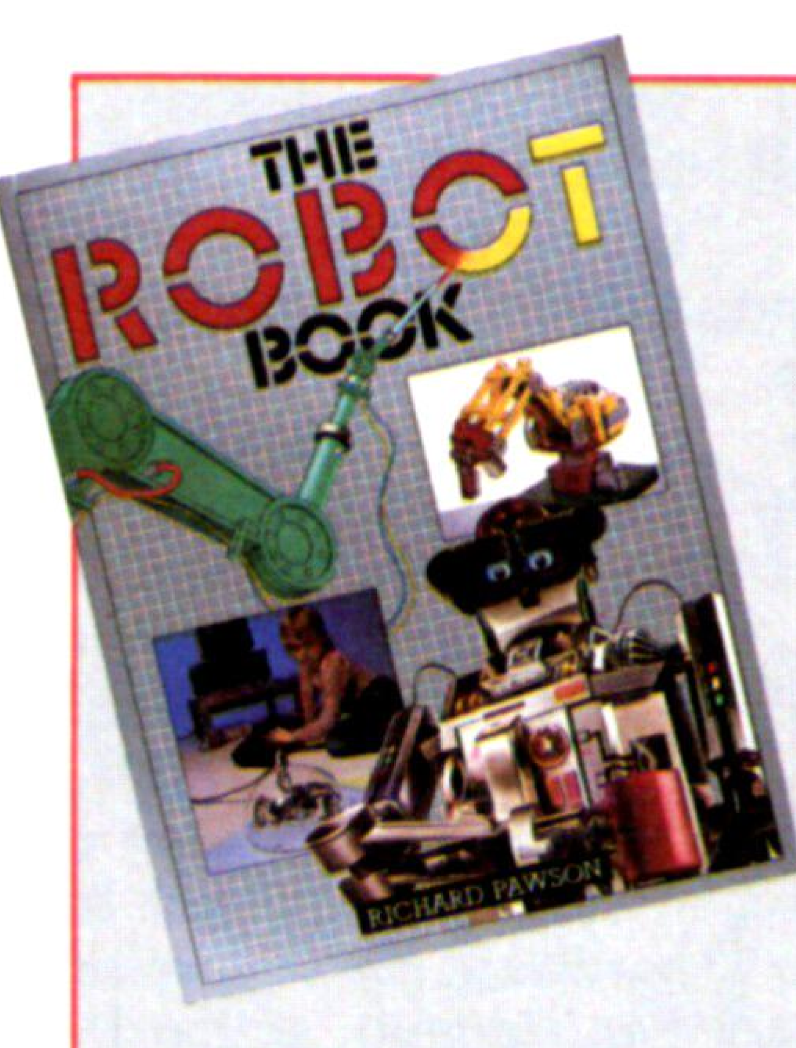
Lingüística aplicada

La tabla muestra una lista de aplicaciones, y los lenguajes más adecuados, por orden de preferencia:

Aplicación	Lenguajes
Análisis estadístico	FORTRAN, BASIC, PASCAL
Control de stocks	COBOL, PASCAL, BASIC
Contro de brazo-robot	FORTH, C, ensamblador
Juego de aventuras	PROLOG, C, BASIC
Sistema experto	LISP, PROLOG, LOGO
Educación	LOGO, PROLOG, PASCAL

Copia impresa

En los últimos años la literatura informática ha constituido un auténtico "boom" en la industria editorial. En estos momentos es posible hallar libros que versen sobre las áreas más desconocidas de la informática. He aquí una selección de títulos que informa de la variada gama de temas que abarca esta disciplina. Algunas obras abordan en profundidad materias que hemos cubierto a lo largo de este curso



El libro de los robots

The robot book, libro con una presentación de gran colorido, cubre los diversos aspectos del control por microordenador y la robótica. La historia de la robótica, las aplicaciones industrial, educativa y doméstica de los robots, su funcionamiento y proyectos de construcción prácticos constituyen las cuatro partes principales del libro. Las tres primeras proporcionan un tratamiento teórico del tema, mientras que la última sección incorpora este conocimiento en planes para que el usuario pueda construir sus propias máquinas. Se utilizan piezas de Lego y Fischertechnik y se dan instrucciones completas para conectar los proyectos terminados en interface con su micro. La última sección amplía algunos de los proyectos emprendidos en nuestro apartado de *Bricolaje*, detallando la construcción, entre otras cosas, de robots para jugar a las damas y repartir la baraja, dispositivos que caminan y brazos estáticos. Todos están bien presentados, con diagramas fáciles de seguir y explicaciones completas.

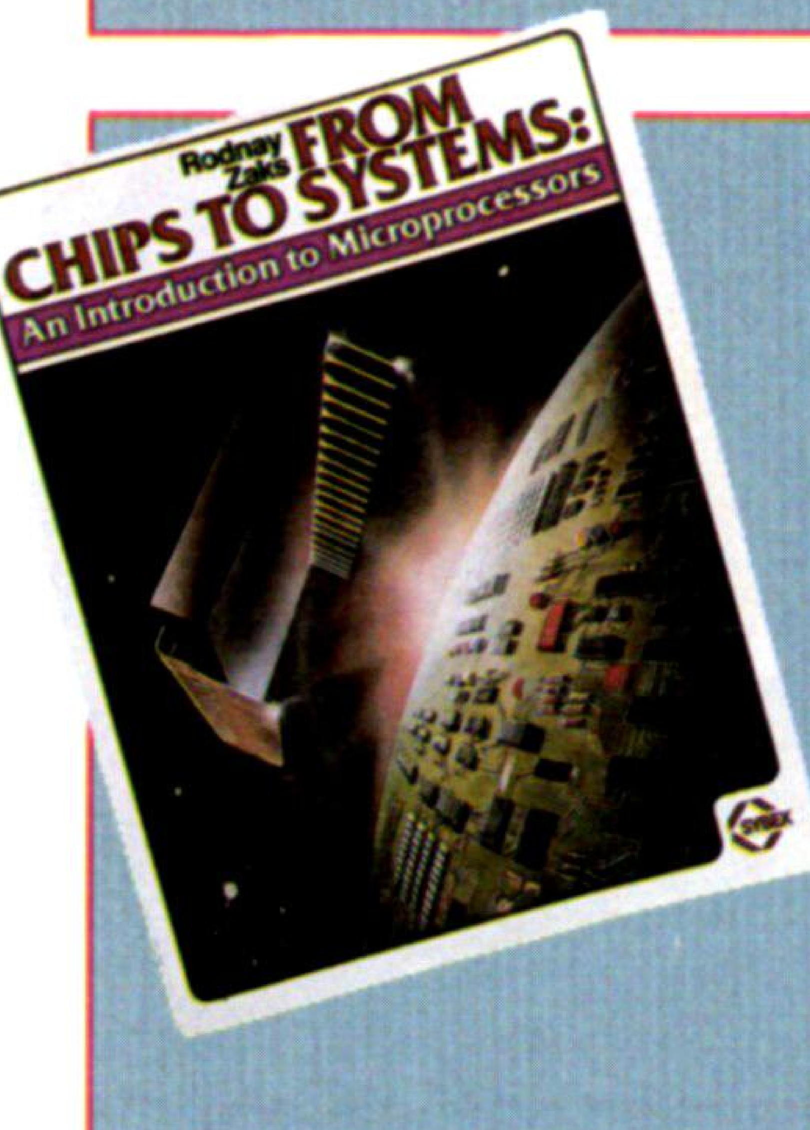
Título: *The robot book*
Autor: Richard Pawson
Editado por: Windward

Guía de la inteligencia artificial para el autostopista

The hitch-hiker's guide to artificial intelligence es una introducción amable e informativa a la inteligencia artificial y desarrolla muchos de los temas tratados en nuestra serie dedicada a ella. Solución de problemas, lenguaje natural y creatividad por ordenador son algunos de los temas que se tratan en los diez capítulos, profusamente ilustrados con didácticos programas en BASIC. Éste es uno de los aciertos del libro, dado que usted no necesitará poseer conocimiento alguno sobre los lenguajes específicos de la AI.

Esta "Guía de la inteligencia artificial para el autoestopista" se encuentra disponible para usuarios del BBC Micro, Apple II, IBM PC y la gama de micros Amstrad.

Título: *The hitch-hiker's guide to artificial intelligence*
Autores: Richard Forsyth y Chris Naylor
Editado por: Chapman and Hall/Methuen



De los chips a los sistemas

From chips to systems reúne en sus páginas todos los aspectos fundamentales del hardware de ordenador, disipando gran parte del misterio que nos impide aventurarnos en el tema. Dando por sentado sólo un conocimiento limitado de informática y electrónica, Rodnay Zaks esboza el funcionamiento de los componentes individuales y detalla cómo se interconectan para conformar sistemas completos. Abordando desde los principios del diseño de microprocesadores hasta la conexión en interface, Zaks analiza luego aplicaciones de microprocesador sencillas, tales como un controlador de motor de cassette, un convertidor analógico/digital y un controlador para horno de microondas.

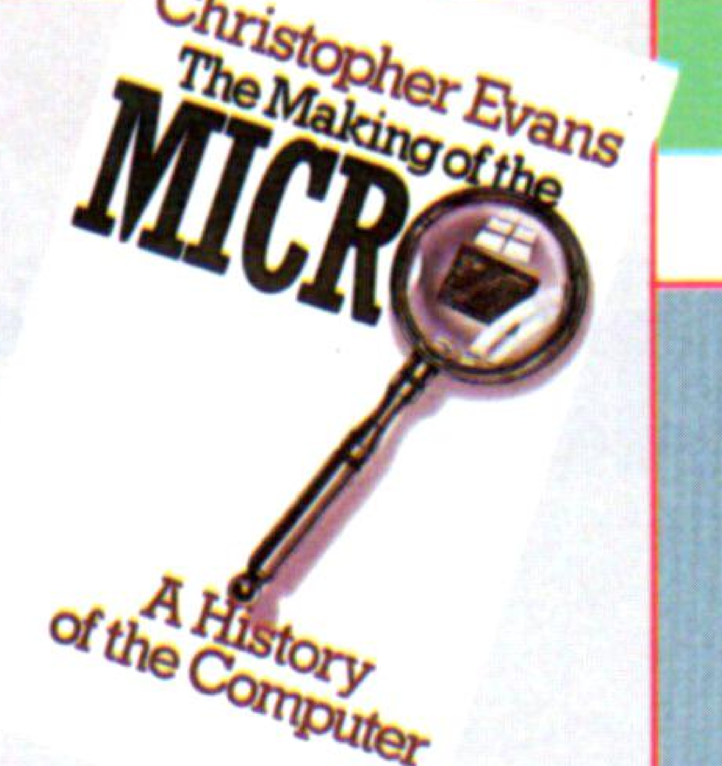
Título: *From chips to systems*
Autor: Rodnay Zaks
Editado por: SYBEX

El manual del "hacker"

The hacker's handbook llegó a los titulares de la prensa británica cuando la policía, temiendo la revelación de información confidencial, insistió en leer el libro antes de que fuera publicado. (Recordemos que se denomina *hacker* [asaltante] al usuario que accede ilícitamente a los ordenadores centrales utilizando ordenadores personales o modems.) Pero al *hacker* experimentado le dice poca cosa que ya no sepa, y al "novato" sólo le proporciona indicaciones vagas. El libro ofrece una detallada reseña de los dispositivos de comunicación, cómo están implementados en diversos sistemas y qué equipo se necesita para iniciarse como "asaltante electrónico". Contemplado bajo este prisma, *The hacker's handbook* es una valiosa guía para redes de datos, desde tablones de anuncios hasta las grandes bases de datos públicas, y, en consecuencia, es recomendable para quien se interese por utilizar un modem... aun cuando no sea para introducirse en los ordenadores de la OTAN.

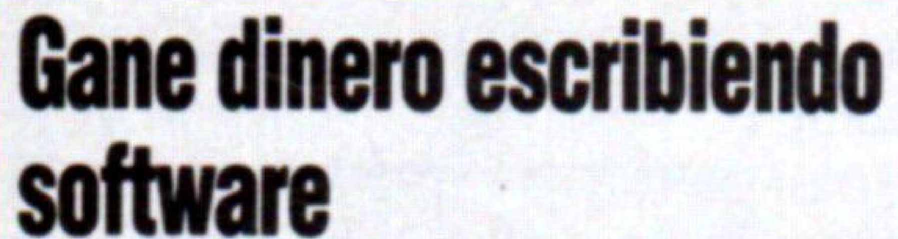
Título: *The hacker's handbook*
Autor: Hugo Cornwall
Editado por: Century

La realización del micro



Al igual que *The soul of a new machine*, de Tracey Kidder, *The making of the micro* proporciona una apasionante información sobre el desarrollo de los ordenadores, en esta ocasión desde la perspectiva histórica. La obra de figuras como Babbage, Von Neumann y Pascal se analizan con todo detalle, y algunas de ellas cobran más vida aún en virtud de la narración de sus asuntos personales.
Un punto en el cual este libro hace hincapié implícitamente es cómo son realmente los ordenadores antiguos, y cómo hemos llegado a darlos por sentado sin rendir verdaderos honores a sus pioneros. Pocas personas sabrán, por ejemplo, que Vannevar Bush realizó una mejora trascendental al reemplazar los elementos mecánicos de su "analizador diferencial" por tubos termiónicos. Esta obra de Christopher Evans constituye una interesante, aunque breve, incursión en los antecedentes de la nueva tecnología.

Título: *The making of the micro*
Autor: Christopher Evans
Editado por: Oxford University Press



Gane dinero escribiendo software

Se han escrito innumerables libros dirigidos a aspirantes a programadores que sueñan con escribir un programa que les haga ganar una fortuna. *Writing software for profit*, sin embargo, pretende señalar los escollos ocultos de esta búsqueda, sugiriendo algunos puntos a tener en consideración al escribir la primera línea de código. En su obra, A. J. Harding da una idea general sobre la industria del software, ilustrando la clase de programación y la presentación esperadas: un aspecto del negocio que con frecuencia soslayan otras publicaciones que tratan este tema.

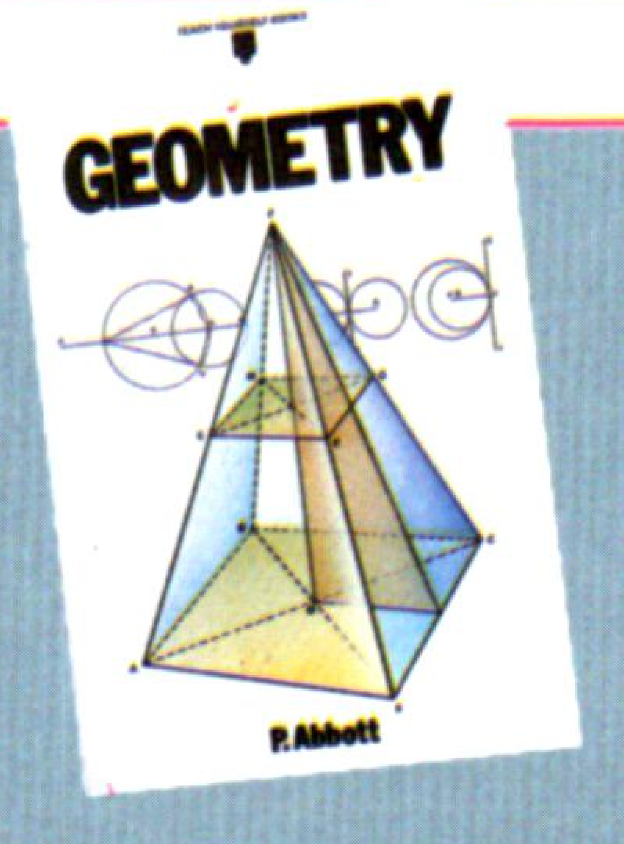
Título: *Writing software for profit*
Autor: A. J. Harding
Editado por: Virgin Books

Reinventando al hombre

Reinventing man se ha convertido en una suerte de obra de referencia obligada para quienes se interesen por la robótica. Sin embargo, no proporciona demasiados detalles técnicos acerca de los robots. Intenta más bien repasar la historia del "hombre mecánico", comparándola con el desarrollo actual y probable y abatiendo varias falacias a lo largo de sus páginas.

Los autores explican de forma detallada cómo la tecnología actual puede conducir al hardware y el software que probablemente se necesitará para desarrollar los robots tan mitificados por los amantes de la ciencia-ficción, y predicen las limitaciones que es probable que tenga esta tecnología. Quien esté interesado en la inteligencia artificial y la robótica encontrará este libro absorbente, tanto por su riqueza de información como por el apasionante tema que aborda.

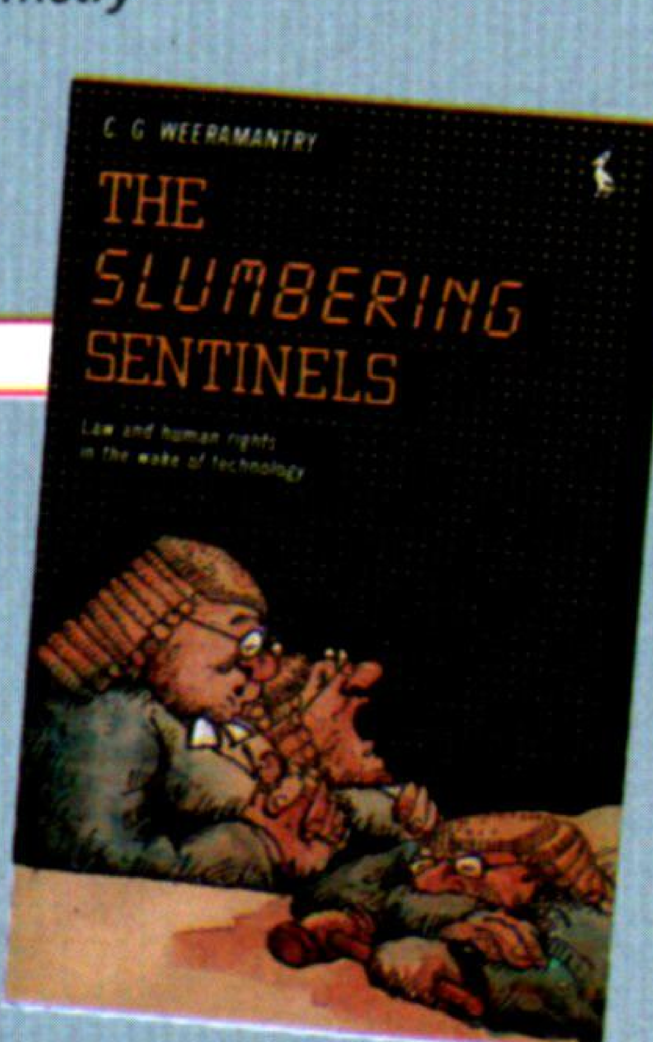
Título: *Reinventing man*
Autores: Igor Aleksander y Piers Burnett
Editado por: Kogan Page



Aprenda geometría por sí mismo

Una de las dificultades con que tropiezan muchos programadores cuando empiezan a abordar proyectos más ambiciosos es la falta de teoría matemática o algebraica, el conocimiento de la cual puede ser especialmente útil para crear métodos de codificación nuevos y más compactos. *Teach yourself geometry*, al igual que otros libros de la serie, ofrece una base sólida en su tema, complementada con varios ejercicios, cuyas respuestas se incluyen al final. Es especialmente útil si usted desea hacer mucha programación de gráficos y constituye un significativo ejemplo de un tipo de literatura que, aunque no dirigida específicamente a los usuarios de ordenadores, puede ser de suma utilidad.

Título: *Teach yourself geometry*
Autor: P. Abbott
Editado por: Teach Yourself Books/Hodder and Stoughton



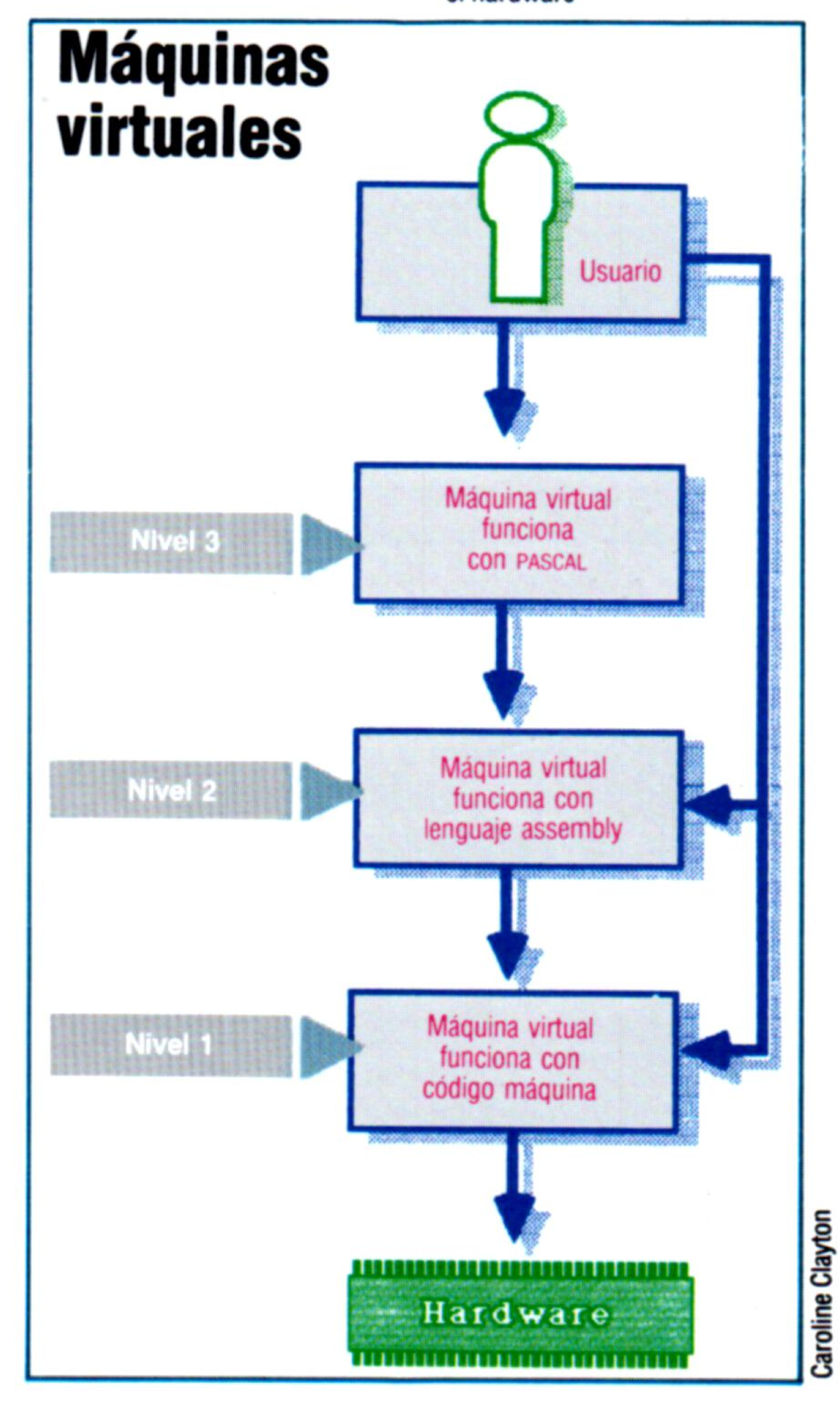
Los centinelas dormidos

The slumbering sentinels aborda un aspecto esencial de la nueva tecnología que está surgiendo: cómo la enfoca, la encuadra y regula la ley. Asimismo, se centra en dilemas morales planteados por los desarrollos tecnológicos y su efecto sobre los derechos del individuo. El libro no versa exclusivamente sobre ordenadores, pero aborda el tema con inquietante frecuencia, ilustrando la insuficiencia del marco constitucional actual de los diferentes países para hacer frente a situaciones tales como el almacenamiento y la revelación de información confidencial, evidenciando su renuencia a abordarlos, o al menos eso es lo que parecería. Del aspecto legal de la informática se habla raramente, y es su incursión en el campo del derecho lo que convierte a estos "centinelas dormidos" en un estudio interesante para quien esté comprometido en la aplicación (especialmente comercial) de la nueva tecnología.

Título: *The slumbering sentinels*
Autor: C. G. Weeramantry
Editado por: Penguin (Pelican Originals)

Tradúzcamelo

Niveles de compromiso
Desde el punto de vista del programador, se puede considerar el ordenador como tres "máquinas virtuales". A un nivel 3, el ordenador del ejemplo funciona en PASCAL. A nivel 2, funciona en assembler, y a nivel 1 el código máquina directamente se enfrenta con el hardware



Este capítulo final sobre programación del 68000 está dedicado al estudio de la operativa del ensamblador

A lo largo de esta serie hemos empleado con frecuencia las instrucciones a nivel ensamblador para ilustrar las características del microprocesador 68000. En nuestros análisis supusimos que el ordenador obedece estas instrucciones según van apareciendo en su forma ensamblador. La máquina interpreta de hecho un código binario que representa las instrucciones ensamblador de nivel más alto, siendo tarea del ensamblador el *traducir* estas instrucciones a código máquina.

Sin embargo, es conveniente considerar la ejecución de nuestros programas como si la máquina los ejecutara a nivel fuente. Este nivel puede ser muy alto (p. ej., los paquetes de aplicación como el *WordStar*) o ligeramente inferior (como los programas en PASCAL) o muy bajo, casi equiparable al nivel de los programas de ensamblador. Esto explica el enfoque a varios niveles de nuestra visión de la máquina, que se compone de estratos discretos o niveles que se convierten en máquinas *virtuales*, ejecutoras de un determinado lenguaje de programación, como se ilustra en el diagrama (derecha).

El modo como un programa funciona a cualquier nivel es uno de éstos:

- *Traducción*: Un programa que está a un nivel se traduce al nivel inferior de la máquina. Por ejemplo, podemos traducir un programa PASCAL de nivel 3 en ensamblador para ser tratado a nivel 2 por un compilador.
- *Interpretación*: Un programa en un nivel es interpretado por un programa (denominado *intérprete*, claro está) que funciona en una máquina de nivel inferior. Por ejemplo, la máquina interpreta los modelos binarios de bits que representan las instrucciones para operaciones de máquina (o nivel de registro). Igualmente, el nivel 2 puede interpretar también programas en PASCAL desde el nivel 3.

Lo que más nos interesa aquí es el proceso de traducción. Hay muchas formas de traducción que van desde los compiladores hasta los ensambladores, pero todas tienen una cosa en común: todas toman instrucciones de alto nivel y proporcionan las correspondientes instrucciones a nivel inferior. Las instrucciones fuente originales no se precisan entonces, y en cierto sentido son "tiradas a la papelera" por el traductor (sin embargo, un intérprete todavía tendría necesidad del programa fuente original).

El objetivo del ensamblador es traducir las instrucciones de ensamblador en alto nivel a código binario ejecutable por la máquina. Por ejemplo:

```
MOVE.W D3,D5
```

se traduciría así:

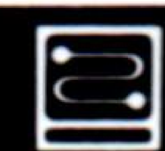
```
0011 101 000 000 011
```

El 0011 corresponde a "mover una palabra" (MOVE.W); 101 000 corresponde a "al D5"; y 000 011 corresponde a "desde el D3".

Pero el trabajo a nivel de codificación de bits es extraordinariamente tedioso y propenso a errores, por lo que como mínimo necesitamos expresiones mnemotécnicas que nos permitan recordar las instrucciones y los objetos de datos. Por ejemplo, MOVE.W TOTAL,D4 significa que nos podemos referir a posiciones de memoria mediante nombres simbólicos que nos indiquen el significado del contenido de esa posición (TOTAL en este ejemplo).

Los tipos de error que podríamos cometer si codificáramos los bits a mano serían, entre otros:

- Equivocar los códigos de instrucción (opcodes).
- Direccionamiento absoluto incorrecto.
- Asignar un número inexacto de bytes por instrucción.



Esto quiere decir que no hablaremos siquiera del caso en que haya que hacer una traducción manual que supere media página de instrucciones. De aquí la importancia desempeñada por el ensamblador en la traducción del código fuente al código objeto.

El proceso assembly

El proceso assembly se inicia cuando el ensamblador lee las sentencias fuente en el archivo de texto fuente (texto escrito en ASCII) y produce un archivo de listado (o lo imprime) más un archivo objeto (o lo carga en la máquina). Nuestro archivo Formato Listado Assembly, que ilustra los elementos de una traducción, es un programa que ejecuta un cálculo aritmético en los elementos de una tabla.

Varios son los aspectos dignos de nota en el listado. Primero, usted establece una lista de contenidos de las posiciones y después el resto de la línea continúa con el código fuente original que sigue a un número insertado de sentencia. La información de errores se referirá al número de sentencia. Si la línea 14 contiene un error, se imprimirá E en esa línea y se hará una referencia a esa línea en la sección de información de errores del listado.

El formato del archivo binario es interesante por varias razones. Ante todo, el archivo ha de contener la información binaria en forma codificada, así como información del cargador para poder cargar correctamente el código binario en la memoria. La manera de conseguir esto es almacenando la información de direcciones y contenidos en binario codificado en hexa, por lo que el cargador debe conocer el formato binario. El formato exacto depende del ensamblador, pero la compatibilidad del cargador es esencial.

La última parte del listado es la impresión de la tabla de símbolos. Esta tabla proporciona los valores numéricos asignados a los niveles declarados en el programa. Por ejemplo, INPUT (definido en la sentencia 25) tiene valor 1024 (hexa) y está referenciado en la sentencia 10. Son todos detalles casi triviales en este pequeño programa de ejemplo, pero en un programa algo más extenso se convierten en información esencial para la depuración de errores.

El destino de la salida depende de donde esté funcionando el ensamblador (naturalmente, el ensamblador es un programa como otro cualquiera). Si, por ejemplo, estamos usando un método de *ensamblador cruzado*, el ensamblador funciona en un sistema huésped como el Unix, y el listado y la información binaria estarán en archivos. El archivo binario ha de ser cargado en el 68000 de destino antes de poder ejecutar el programa.

Otro modo de hacer esto es ejecutar el ensamblador en la máquina destino, si ésta tiene al menos algún tipo de sistema de gestión de archivos, aunque sea rudimentario. A veces los códigos binarios se cargan directamente en la memoria con el assembly destino y los listados pueden ser impresos directamente si está activada la impresora.

Los mecanismos del ensamblador

Los mecanismos que emplea un ensamblador nos dan idea de los problemas que pueden presentarse (¡que inevitablemente se presentarán!). Por lo general, el proceso de assembly es una organización de *dos pasos*. En el primer paso, el ensamblador:

- Decodifica los mnemotécnicos assembly (MOVE, ADD, MULS y demás).
- Cuenta los bytes de dirección (para poder calcular las direcciones).
- Construye una tabla de símbolos (que nos dice los valores asignados a cada símbolo).
- Descarta cualquier error de sintaxis (p. ej., ADDQQ es una instrucción ilegal).

Cuando el ensamblador lee la sentencia 9 en el primer paso por el programa *Formato listado assembly*, el contenido de etiqueta de la posición será 41F8. Que corresponde a un "movimiento rápido" a D0 en el modo inmediato con una constante de 12 (desde LENGTH en la tabla de símbolos).

Para la sentencia 10, el ensamblador puede establecer la posición 1002 con 41F8 para la instrucción LEA, pero no puede establecer todavía la dirección de INPUT. Así, la posición 1004 se deja vacía hasta el segundo paso cuando se conoce la dirección de INPUT.

En el segundo paso, el ensamblador puede sustituir las direcciones ahora conocidas dondequiera que se empleen dentro del programa, y dar salida al código binario y listado del programa. Pueden también darse errores que depurar, pero por lo general se atienden en el primer paso.

Formato listado assembly

```
POS      OBJETO                    SENT                          SENTENCIA FUENTE
                                   1   *Esta sentencia toma cada elemento de una tabla
                                   2   *y lo convierte en
                                   3   *                    INPUT[I]:=2*INPUT[I]+3
                                   4
                                   5
001000                             6              ORG $1000
         =000C                     7   length:    equ 12
                                   8
001000   700C                      9   start:     moveq     #length,d0
001002   41F8  1024                10             lea       input,a0
001006   4244                      11             clr       d4
001008   4243                      12             clr       d3
                                   13
00100A   3610                      14  loop:      move.w    (a0),d3
00100C   C7FC  0002                15             muls      #2,d3
001010   D67C  0003                16             add.w     #3,d3
001014   30C3                      17             move      d3,(a0)+
001016   D843                      18             add       d3,d4
                                   19
001018   5340                      20             subq      #1,d0
00101A   6600  FFEE                21             bne       loop
00101E   31C4  103C                22             move      d4,sum
001022   4F40                      23             trap      #0
                                   24
001024   0001  0002  0003  0004    25  input;     dc.w      1,2,3,4,5
         0005
00102F   0006  0007  0008  0009    26             dc.w      6,7,8,9,10
         000A
001038   000B  000C                27             dc.w      11,12
                                   28
00103C   0000                      29  sum:       dc.w      0
                                   30
                                   31             end
No se encontraron errores en este ensamblaje
```

SÍMBOLOS	DEFN	VALOR	REFERENCIAS
INPUT	25	1024	10
LENGTH	7	000C	9
LOOP	14	100A	21
START	9U	1000	
SUM	29	103C	22

Podemos emplear asimismo el ensamblador como una útil calculadora, creando las expresiones simbólicas que calculan el valor de un operando. Por ejemplo, estas líneas introductorias de un listado assembly:

```
LENGTH   DC.W      (START-ARRAY)*4
ARRAY    DC.W      1,2,3,4,5
START    MOVE.W    LENGTH,D3
```

Aquí necesitamos calcular la longitud de la tabla en bytes (START-ARRAY multiplicada por 4), para emplearlo en el programa (para poner 3 en el registro de datos en este caso). Si más tarde cambiamos en algún momento la longitud de la tabla dentro del programa en ensamblador, LENGTH se restablecerá automáticamente.

Ayudas de documentación

Otra ayuda que proporciona el ensamblador está en las áreas de documentación. Es posible agregar comentarios a nuestros programas que orienten a los lectores sobre el significado de las instrucciones, y también podemos incluir detalles sobre registros convenidos y diseño del programa, por ejemplo. Véase el siguiente listado:

```
*Programa para entrar una cadena de caracteres
 y verificarla
*El texto es entrado a través de ACIA y verificado
*A. Gómez - Dic 85
*
*Registros convenidos
*
*Registros de direcciones
*        A1 puntero para entrar cadena
*        A2 puntero para almacenar teclado
*        A3 puntero para carácter actual
*Registros de datos
*        D0 entrada carácter
*        D1 salida carácter
*
START JSR INIT           inicializa la E/S
      JSR READSTRING     lee cadena entrada
      JSR VALIDATE       y la comprueba
      JSR SIGNAL         señal si es correcta
      BRA START          bucle indefinido
```

Leyendo estos comentarios nos informamos sobre el programa, su estructura, los registros convenidos. Probablemente sería pedir demasiado si se exigieran aún más detalles a este nivel, pero es claro que el lector puede descender a otro nivel y mirar los comentarios contenidos en el código de la subrutina si necesita más información.

Lo importante es que el programa quede razonablemente bien documentado con el empleo de comentarios.

Directivas del ensamblador

También el assembler nos permite otras ayudas a la programación en el área general de las *directivas del ensamblador*. Pueden ser meras ayudas documentales como TTL o "directiva del título", que ordena la impresión de un determinado título en cada página del listado, o un medio para pasar la información al ensamblador. Por ejemplo, puede que deseemos señalar el final de nuestro texto de entrada para que el ensamblador comience su ensamblaje. Nuestro módulo del programa contendrá las siguientes líneas:

```
*Los comentarios de documentación
*Con nuestro nombre y fecha por lo menos
      TTL   Éste es mi programa
      ORG   $1000   *una instrucción de origen

START MOVE D1,D2    *y algunas instrucciones
                    *y otras y otras más
      END           *y la directiva de final
```

Las más importantes directivas del ensamblador se resumen en la tabla de más abajo.

Formato del ensamblador

No es de extrañar que los ensambladores construidos por distintos fabricantes difieran en detalles en cuanto a su operativa y formato. No obstante, la mayoría de los ensambladores se ajustan a una estructuración fuente semejante a la que sigue. Primero, los comentarios pueden ponerse empleando el signo * como primer carácter de la línea, o después de una instrucción si entre ésta y el comentario media un espacio como mínimo. Así se permiten estas dos líneas:

```
*Esto sólo es un comentario
START  ADD  D1,D2 *y esto, otro
```

Lo que constituye una instrucción debe naturalmente definirse, y consta de tres campos opcionales:

- *Campo de etiqueta*: Cada nombre empleado como una etiqueta debe comenzar con un carácter alfabético y tendrá de longitud total menos de treinta caracteres alfanuméricos. Se notará que si se omite este campo se ha de insertar al menos un espacio.
- *Campo del opcode*: Se empleará una del conjunto de instrucciones del 68000.
- *Campo del operando*: Los operandos de instrucción legales deben usarse después de un espacio al menos tras el campo del opcode. Pero en el campo del operando no deben existir espacios aun en el caso de emplear dos operandos.

He aquí algunos ejemplos de sentencias no permitidas en ensamblador:

Tabla de directivas del ensamblador

ORG	Abreviatura de "origen". Especifica la dirección de inicio del código
RORG	Vale lo dicho para ORG, pero con código relativo del PC
TTL	Pone un título en cada página del listado
END	Fin del texto para ensamblar
EQU	"Igual a" (*equate*): asigna un valor al símbolo especificado (p. ej., LENGTH EQU 10+50*ARRAY1, donde LENGTH vale 610 suponiendo que ARRAY1 vale 12)
DC	"Declarar Constante": asigna un valor constante (byte, palabra o longitud palabra larga). P. ej., PARAM DC.W 12,23 hace a PARAM una palabra que contiene 12 y PARAM+2 es una palabra que contiene 23
DS	"Declarar eSpacio": reserva un área de datos no inicializados (p. ej., STACK DS.B 100 establece un espacio de 100 bytes llamado STACK)
LLEN	Establece la longitud de línea
PAGE	Envía un salto de página durante el listado

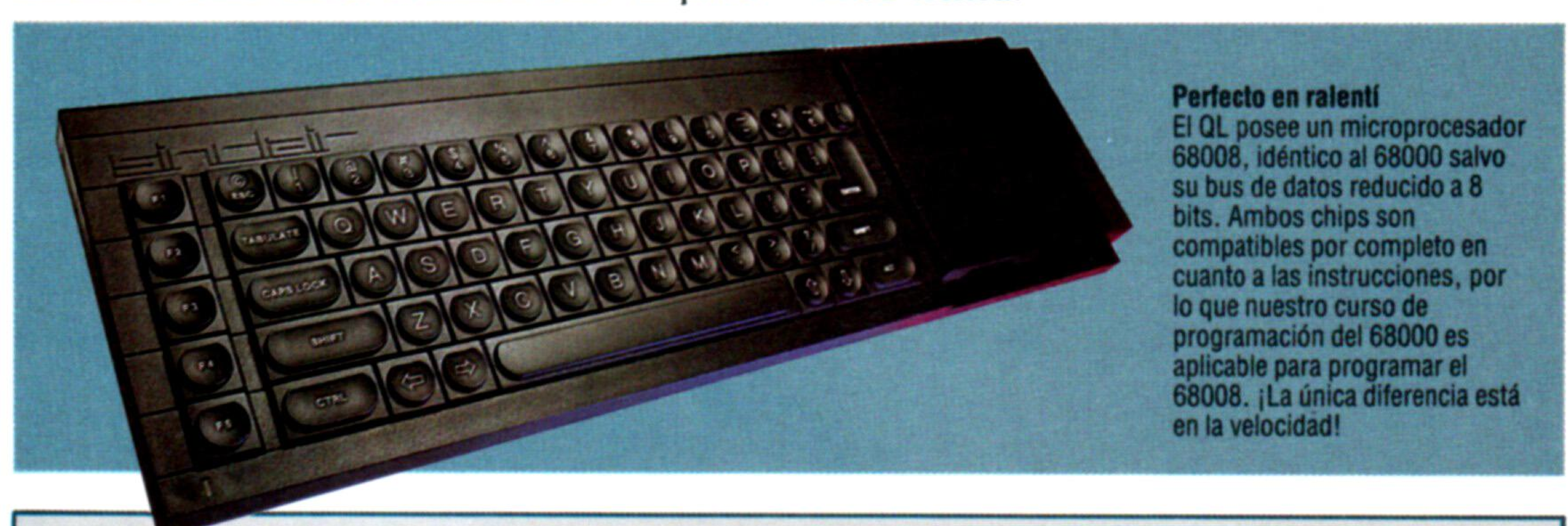


```
Esto no es un comentario del todo pues falta el *
MOVE      D1, D2      *no debe haber espacios
                      *entre los operandos
MOVE    8TOIT,D5      *los nombres han de
                       comenzar
                      *con una letra y ADEMÁS
                       mediará
                      *al menos un espacio entre
                      *instrucción y operando
RTS                   *correcto
RTS       D1          *esto ya no (pues D1 aquí
                      *no tiene sentido)
   *fin de ejemplo erróneo con comentarios verdaderos!
```

Digamos, para acabar, algo sobre la representación alfanumérica. Los números se representan en el ensamblador como valores decimales si no van precedidos del signo $. Así, 1234 es decimal pero $1234 es hexadecimal. Para las letras ASCII los caracteres se encierran entre comillas sencillas. Por ejemplo:

'Que pena, este curso se acaba!'

Acabamos de examinar el empleo del ensamblador como una herramienta de programación, ya no sólo como un medio de introducir codificación en el 68000. Las facilidades ofrecidas son bastante buenas especialmente con la facilidad macro y las bastante buenas directivas del ensamblador, así como los mensajes de error. Las ayudas de documentación también han sido dignas de resaltar, ¡dado que nunca han de olvidarse en favor de otras personas que sin duda habrán de leer los programas que usted escriba!



Perfecto en ralentí
El QL posee un microprocesador 68008, idéntico al 68000 salvo su bus de datos reducido a 8 bits. Ambos chips son compatibles por completo en cuanto a las instrucciones, por lo que nuestro curso de programación del 68000 es aplicable para programar el 68008. ¡La única diferencia está en la velocidad!

Facilidad MACRO

Justamente la facilidad MACRO es un método para sustituir textos en un programa fuente que se ha revelado como una herramienta poderosa de programación en grandes programas. Observemos que donde se especifica una "macro" se sustituirá el texto macro y el anteriormente definido se rechazará. Se puede, por ejemplo, definir así una macro ERROR:

```
ERROR     MACRO
          MOVE.B      #3,ERRORLOC
          JSR         SIGNAL
          ENDMACRO
```

Finalmente, la palabra ERROR producirá, en tiempo de ensamblaje, la siguiente serie de instrucciones:

```
MOVE.B    #3,ERROR.LOC
JSR       SIGNAL
```

Ahora podemos ampliar esta facilidad para incluir el empleo de parámetros a cada llamada si lo precisamos.

Mire este ejemplo:

```
          TTL         Macro ejemplo
          ORG         $1000
ADDON     MACRO
          ADD         \1,D2
          ADD         D2,\2
          ENDM

          ADDON       AVAL,SUM
          ADDON       BVAL,SUM
          ADDON       SUM,TOTAL
AVAL      DC.W        0
BVAL      DC.W        0
SUM       DC.W        0
TOTAL     DC.W        0
          END
```

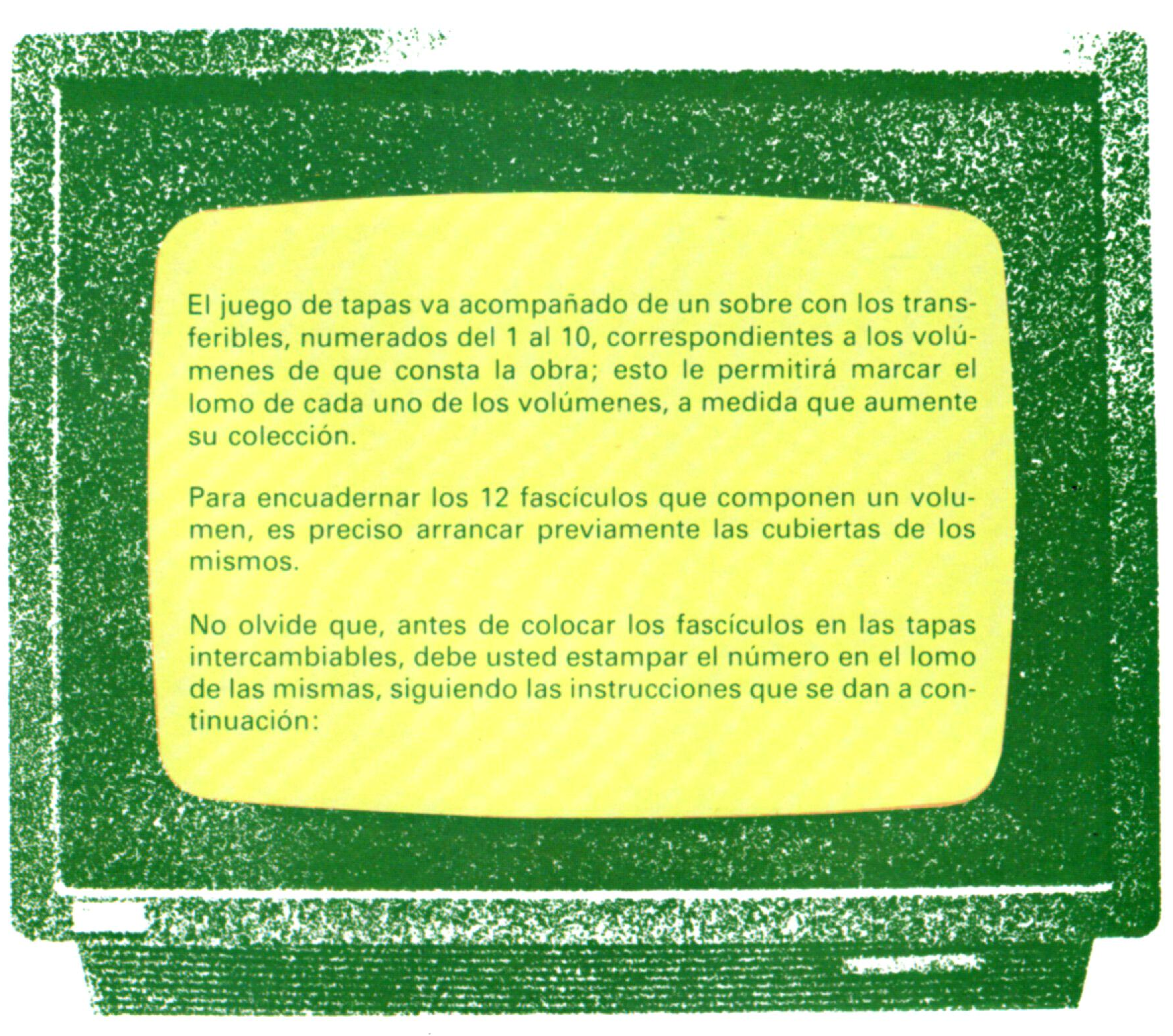
Observará que\1 y\2 se refieren a los parámetros primero y segundo. En tiempo de ensamblaje, esto producirá:

```
3               ORG         $1000
4
5    ADDON      MACRO
6               ADD         \1,D2
7               ADD         D2,\2
8               ENDM
9
10              ADDON       AVAL.SUM
11+             ADD         AVAL,D2
12+             ADD         D2,SUM
13              ADDON       BVAL,SUM
14+             ADD         BVAL,D2
15+             ADD         D2,SUM
16              ADDON       SUM,TOTAL
17+             ADD         SUM,D2
18+             ADD         D2,TOTAL
19
20   AVAL       DC.W        0
21   BVAL       DC.W        0
22   SUM        DC.W        0
23   TOTAL      DC.W        0
24              END
```

Recuerde que los signos + después de los números de sentencia indican las líneas insertadas por el macroprocesador.



Con el próximo fascículo se pondrán a la venta las tapas correspondientes al décimo volumen.



El juego de tapas va acompañado de un sobre con los transferibles, numerados del 1 al 10, correspondientes a los volúmenes de que consta la obra; esto le permitirá marcar el lomo de cada uno de los volúmenes, a medida que aumente su colección.

Para encuadernar los 12 fascículos que componen un volumen, es preciso arrancar previamente las cubiertas de los mismos.

No olvide que, antes de colocar los fascículos en las tapas intercambiables, debe usted estampar el número en el lomo de las mismas, siguiendo las instrucciones que se dan a continuación:



1. Desprenda la hojita de protección y aplique el transferible en el lomo de la cubierta, haciendo coincidir los ángulos de referencia con los del recuadro del lomo.

2. Con un bolígrafo o un objeto de punta roma, repase varias veces el número, presionando como si quisiera borrarlo por completo.

3. Retire con cuidado y comprobará que el número ya está impreso en la cubierta. Cúbralo con la hojita de protección y repita la operación anterior con un objeto liso y redondeado, a fin de asegurar una perfecta y total adherencia.

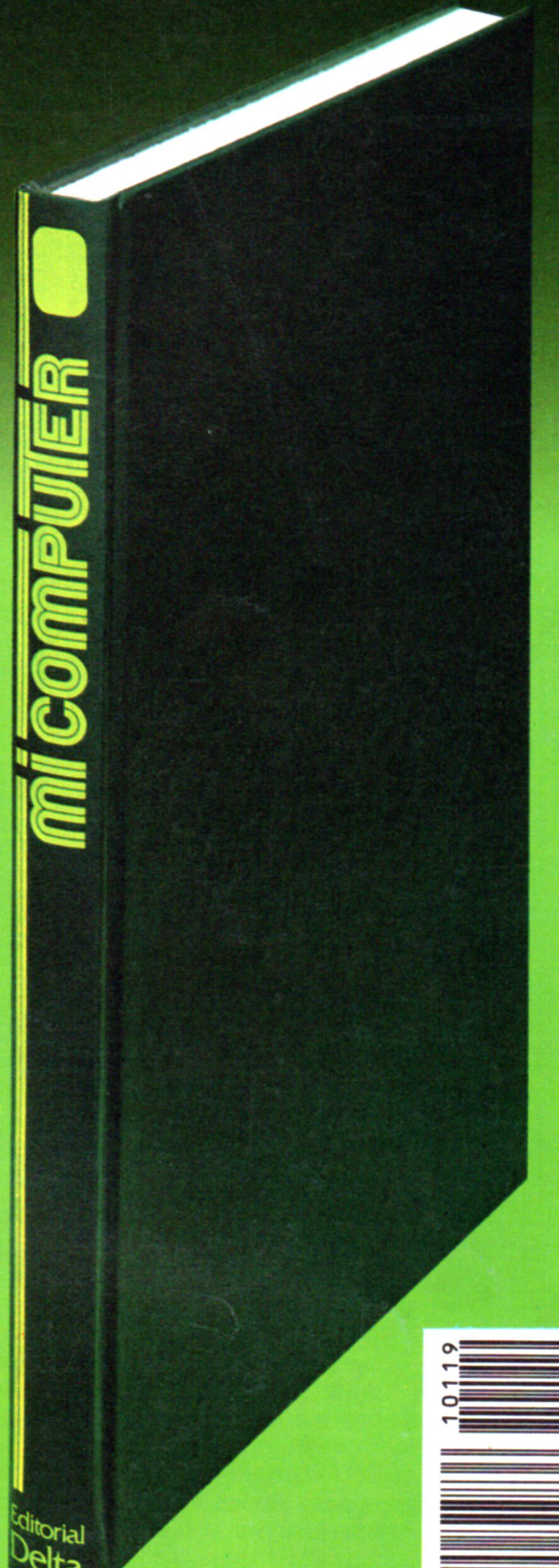
Editorial Delta

Cada sobre de transferibles contiene una serie completa de números, del 1 al 10, para fijar a los lomos de los volúmenes. Ya que en cada volumen sólo aplicará el número correspondiente, puede utilizar los restantes para hacer una prueba preliminar.

Con el próximo número se ponen a la venta las tapas intercambiables para encuadernar 12 fascículos de

mi computer

Cada juego de tapas va acompañado de una colección de transferibles, para que usted mismo pueda colocar en cada lomo el número de tomo que corresponda





10119
9 788485 822836